Defence Research and Development Canada
Recherche et développement pour la défense Canada

DEFENCE R&D DÉFENSE

Investigation of a Neural Network Implementation of a TCP Packet Anomaly Detection System

M. Dondo and J. Treurniet

Defence R&D Canada – Ottawa
TECHNICAL MEMORANDUM
DRDC Ottawa TM 2004-208
May 2004

Canada

Investigation of a Neural Network Implementation of a TCP Packet Anomaly Detection System

M. Dondo

J. Treurniet

Defence R&D Canada - Ottawa

Technical Memorandum

DRDC Ottawa TM 2004-208

May 2004

106-03819/

© Her Majesty the Queen as represented by the Minister of National Defence, 2004

© Sa majesté la reine, représentée par le ministre de la Défense nationale, 2004

Abstract

We present the design and implementation of an artificial neural network (ANN) system of multi-layer perceptron classifiers to detect suspicious TCP traffic at a single packet level. The advantage to using ANNs for the detection of attacks is that they do not only rely on attack signatures, as in many common signature-based IDSs. Rather they are capable of learning broader definitions of attack attributes. The use of ANNs in this approach also enhances the processing speed where real-time applications require the processing of substantial amounts of data at high speeds. The ANN model was tested on labelled sets of attack data obtained from the DARPA IDS Evaluation. The model was successful in detecting a variety of attacks, including denial of service attacks, probing activity and other suspicious activity. Future work will examine the application of an ANN to sequences of related packets to detect attacks.

Résumé

Nous exposons ici la conception et la mise en oeuvre d'un réseau de neurones artificiel (ANN) formé de classificateurs de perceptrons qui décèlent du trafic TCP suspect au niveau d'un simple paquet. L'avantage de recourir à des réseaux ANN pour la détection des attaques tient au fait qu'ils ne reposent pas exclusivement sur les signatures des attaques, comme les systémes IDS courants à base de signature. Au contraire, ils sont capables d'apprendre des définitions plus larges des attributs des attaques. Les réseaux ANN utilisés dans cette approche accélèrent la vitesse de traitement surtout pour les applications en temps réel qui exigent le traitement de grandes quantités de données à haute vitesse. Le modèle ANN a été testé sur les jeux étiquetés de données pirates obtenues de l'évaluation IDS de la DARPA. Le modèle a bien réussi à détecter un diversité d'attaques, y compris des denis de service, des activités de sondage et d'autres activités suspectes. Dans les travaux à venir, nous examinerons l'application d'un ANN à des séquences de paquets dépendants dans l'espoir de repérer des attaques.

This page intentionally left blank.

Executive summary

The Transmission Control Protocol/Internet Protocol (TCP/IP) suite of protocols are used by the majority of Internet data communications applications. This includes the World Wide Web hypermedia system which uses HTTP (HyperText Transfer Protocol), as well as other common network protocols such as FTP (File Transfer Protocol), SMTP (Simple Mail Transfer Protocol) and Telnet. The widespread use of TCP means that it is likely to be exploited for misuse and various forms of attacks. Malicious behaviour can be perpetrated through the TCP/IP protocols without being blocked by firewalls or detection by intrusion detection systems (IDS) because commonly-used IDSs don't have the ability to recognize new variations of attacks.

Artificial neural networks (ANNs) are capable of learning from previously observed patterns and scenarios. Unlike existing approaches [1], ANN models are capable of classifying large amounts of data without significant computational effort, thus making them suitable for real-time applications. They are easily scalable, and are capable of handling large amounts of data without compromising computational speed [2, 3]. An ANN anomaly detector would be suitable for operational deployment behind the firewall in order to catch what the firewall failed to block. It may also be implemented as a host-based IDS.

In this work, an ANN model was developed that is capable of detecting variations from normal TCP traffic. The attributes of normal and abnormal communication within TCP packets were observed, and the attributes that signify attack and misuse were determined. These attributes were then used in designing and implementing an ANN model capable of automatically classifying and identifying some network-based intrusions at a packet-by-packet level.

The model was applied to the 1999 Defence Advanced Research Projects Agency (DARPA) IDS evaluation data collected by the Lincoln Laboratories at MIT. Of the documented attacks in the DARPA 1999 data, the model was successful in detecting scanning, denial of service and other attack attempts. The flexibility of the model's classifiers was also successfully demonstrated through the use of this data. The IP address classifier, which allows one to define IP domains whose access indicates some form of misuse, was modified and re-trained in the DARPA analysis to account for the simulation network's use of private IP address spaces.

This ANN model handles individual packets, but it is only a first step in the investigation of the application of ANNs to traffic analysis and anomaly detection. Since there are a variety of attacks whose presence may only be determined through observing multiple related packets, future research efforts will focus on enhancing the capabilities of this model to be able to detect malicious events and coordinated

attacks in a series of related packets.

M. Dondo, J. Treurniet; 2004; Investigation of a Neural Network Implementation of a TCP Packet Anomaly Detection System; DRDC Ottawa TM 2004-208; Defence R&D Canada - Ottawa.

Sommaire

La suite de protocoles TCP/IP (Transmission Control Protocol/Internet Protocol) se retrouve dans la majorité des applications de communication de données via Internet. Au nombre de ces applications, il y a le système hypermédia WWW qui fonctionne avec HTTP (HyperText Transfer Protocol), ainsi qu'avec d'autres protocoles courants comme FTP (File Transfer Protocol), SMTP (Single Mail Transfer Protocol) et Telnet. L'étendue de la présence de TCP augmente le risque qu'on s'en serve pour des usages malveillants et diverses autres formes d'attaques. Les comportements malveillants peuvent agir au travers des protocoles TCP/IP à l'insu des pare-feux et des systèmes de détection des intrusions (IDS), parce que la plupart de ces IDS n'ont pas la capacité de reconnatre les nouvelles variantes d'attaque.

Les réseaux de neurones artificiels (ANN) ont les capacités d'apprendre à partir de schémas et de scénarios déjà observés. Contrairement aux approches existantes [1], les modèles ANN parviennent à classifier une grande quantité de données sans véritable travail de calcul, ce qui les rend très bien adaptés aux applications en temps réel. Ils sont faciles à faire évoluer et peuvent se charger d'une quantité considérable de données sans qu'il y ait ralentissement des opérations [2,3]. Étant donné que les techniques ANN ont en général une plus grande vitesse de calcul et peuvent déceler de nouvelles attaques sans l'intervention d'un opérateur, un détecteur d'anomalies ANN devrait bien s'insérer dans le déploiement opérationnel derrière un pare-feu et capturer ce que le pare-feu a laissé passer. On peut aussi l'implanter comme IDS dans un hte.

Dans le présent travail, on a mis au point un modèle ANN en mesure de détecter des variations d'écart par rapport au trafic normal. Les attributs des communications normales et anormales dans des paquets TCP sont observés, et ceux qui semblent correspondre à une attaque ou à une malveillance sont désignés. Ces attributs servent ensuite à concevoir et à implanter un modèle ANN pouvant automatiquement identifier et classifier certaines intrusions de réseau au niveau de chaque paquet.

Ce modèle a été appliqué aux données d'évaluation 1999 du système IDS de la DARPA (Defence Advanced Research Projets Agency), données recueillies par les Lincoln Laboratories du MIT. De toutes les attaques décrites dans les données 1999 de la DARPA, le modèle est parvenu à analyser les denis de service et d'autres tentatives de piratage. La souplesse des classificateurs du modèle a été démontrée sans l'ombre d'un doute. Le classificateur d'adresses IP, qui permet de spécifier des domaines IP dont l'accès laisse présumer une certaine forme de malveillance, a été modifié et remis en apprentissage dans l'analyse DARPA pour tenir compte de l'utilisation par le réseau de simulation d'espaces d'adresses IP privés.

Ce modèle ANN gère aujourd'hui des paquets individuels, mais ce n'est qu'une première étape de l'étude de la mise en oeuvre de réseaux ANN pour analyser le trafic et détecter des anomalies. Comme il existe une grande diversité d'attaques dont la présence n'est souvent repérable qu'en observant plusieurs paquets reliés, les prochains travaux de recherche seront axés sur l'amélioration des capacités de ce modèle à détecter des événements malveillants et des attaques coordonnées dans une série de paquets reliés.

M. Dondo, J. Treurniet; 2004; Investigation of a Neural Network Implementation of a TCP Packet Anomaly Detection System; DRDC Ottawa TM 2004-208; R&D pour la défense Canada - Ottawa.

Table of contents

Abstract i

Résumé i

Executive summary iii

Sommaire v

Table of contents vii

List of tables viii

List of figures ix

1 Introduction 1

2 Background Theory 3

2.1 Transport Control Protocol 3

2.2 Artificial Neural Networks 6

2.2.1 The Neuron 7

2.2.2 The Activation Function 7

2.2.3 Multi-Layer ANNs 9

2.2.4 ANN Training 10

2.2.5 Training Rules 11

3 Packet Anomaly Detection Approach 13

3.1 Attributes 13

3.1.1 IP Addresses 13

3.1.2 TCP Ports 13

3.1.3 TCP Sequence and Acknowledgement Numbers 14

3.1.4 TCP Session Flags 14

3.1.5 Payload Size 15

3.1.6 Composite Attributes 15
3.2 Classifiers 15
3.3 The ANN Packet Anomaly Detection Model 16
4 The Procedure 18
4.1 ANN Training 18
4.2 ANN Testing and Recall 19
5 Results 20
5.1 Model Training and Validation 20
5.2 Classifier Detection Performance for DARPA 1999 Data . . . 21
5.2.1 False Positives 21
5.2.2 Additional DARPA Data Findings 23
6 Concluding Remarks 25
References 27
Annexes 31
A 1999 DARPA Simulation Network 31
B Summary of *IP* Classifier Results 32
C Summary of *Flags* Classifier Results 33
D Summary of *Ports* Classifier Results 36
E Summary of *Sequence* Classifier Results 40
F ACRONYMS 41

List of tables

1 Summary of the TCP session bit flags 5
2 Valid TCP flag-byte combinations. The listed flag bits are set. 15

3 The ANN classifier configurations. SEQ is the TCP sequence number and ACK is the TCP acknowledgement number. 16

4 The number of inputs and number of nodes in the layers of the ANN classifiers. There is one output node for all classifiers. 17

5 Summary of DARPA 1999 *week 5* detects. 22

6 Misclassifications detected by this approach. 23

List of figures

1 Illustration of the TCP data transfer encapsulation process. 3

2 IP and TCP header layouts. 4

3 TCP data packet exchange and data transfer. 6

4 The basic neuron 7

5 Activation functions 8

6 Multi-layer perceptron 9

7 Implementation of the model. 18

8 The training mean square error. 20

A.1 The 1999 DARPA simulation network. 31

1 Introduction

There are two forms of intrusion detection systems(IDS): misuse detection and anomaly detection. Misuse detection IDSs use signatures to detect intrusion attempts [4, 5]. They are effective in identifying known attacks, however they lack the ability to generalize attack signatures and protect the network from slightly modified versions of known network attacks. Anomaly detection techniques detect behaviour that is not considered to be normal and may detect such attacks, however they are generally not trusted as they are considered to have a high degree of false positives. Moreover, these systems require periodic on-line training which can be undermined by incorporating undesired behaviour into the training data [5, 6].

Researchers have applied numerous approaches to the detection of anomalies, including: statistical, fuzzy systems, genetic algorithms, modular programming and Artificial Neural Network (ANN) approaches [1]. There is significant interest in applying ANN methods to IDSs at all levels (host, application and network) [2, 5, 7–12], due to the advantages of ANNs over other approaches.

The number of packets that must be processed in a very short period of time presents a challenge. ANNs are well-known for having fast response times, which makes them suitable for real-time applications where conventional approaches would not produce comparable results at similar speeds [13, 14]. Genetic algorithms and fuzzy systems are some of the latest research areas to intrusion detection. Used on their own, it has been shown that they cannot match the speed, scalability, and precision of ANNs [13–16].

Neural networks are easily scalable [3]; a change in the problem definition can be easily implemented by changing the number of nodes in the ANN model. Unlike conventional algorithms that require substantial algorithm rework when the problem changes, ANNs can be easily scaled at low additional cost [3]. Once trained, ANNs do not need retraining unless the problem definition changes. Statistical approaches usually depend on the underlying behavioural distributions such as Gaussian distribution; ANNs do not make prior assumptions about the data they handle [3].

Over 90% of Internet traffic has been shown to use the Transmission Control Protocol (TCP) [17]. Because of its widespread use and its impressive growth [17, 18], we have chosen to focus our efforts on the detection of anomalous behaviour within TCP traffic. This paper presents the results of the first stage of a larger research program, where a multi-classifier feedforward ANN is used to identify anomalies within individual TCP packets. In a real-time scenario, this is equivalent to analysing each packet as it arrives at the sensor. The ANN is trained using the backpropagation training algorithm, and used to generalize from previously ob-

served TCP traffic header attributes and to extrapolate beyond the training space provided. A backpropagation algorithm was used because it is easy to configure and train. This algorithm has been successfully used in other intrusion detection research [12].

One challenge for ANN IDSs is their ability to identify the source (or type) of an attack [1]. In this work, this challenge is addressed by breaking down the attributes of a packet and using multiple classifiers, identifying the classifier that triggered an alert.

This paper is organised into 6 sections. In Section 2, we give an overview of the TCP protocol and an introduction to ANNs. In Section 3 we examine the attributes of TCP/IP packets whose values may indicate anomalous behaviour and describe how these attributes may be used to detect misuse. We then define the ANN model and form the classifiers that will be used. In Section 4 we show how the ANN is implemented, and in Section 5 we present the results of the model applied to the DARPA 1999 IDS Evaluation Data [19]. We discuss the results and conclude in Section 6.

2 Background Theory

A summary of the TCP protocol and ANNs as applied to this work are presented. Detailed discussions of these topics are available in most of the standard literature [2,20].

2.1 Transport Control Protocol

The TCP protocol is a connection-oriented protocol that enables the reliable transmission of Internet application traffic. Some of the applications that utilize TCP/IP include HTTP(Hypertext Transfer Protocol), FTP(File Transfer Protocol), Telnet, and SMTP(Simple Mail Transfer Protocol). Figure 1 shows the layers of the TCP/IP protocol suite [20] through which transmitted data passes, in this case on an Ethernet network. When an application sends data, it originates at the application layer,

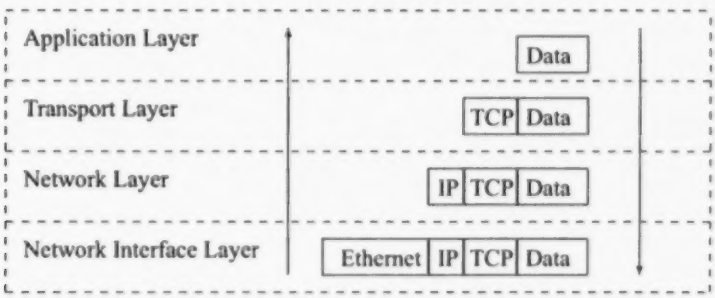

Figure 1: *Illustration of the TCP data transfer encapsulation process.*

then goes to the transport layer, then the network layer, and finally the network interface layer. At each layer, the data is encapsulated with a header containing information about that layer. When the packet is received by a host, the headers are stripped off as the data makes its way from the network interface layer to the application layer. This process is defined in RFC 894 [21]. The TCP and IP protocol headers [20,22] are shown in Figure 2.

The TCP header contains information that is important to the establishment of a reliable connection. The sequence and acknowledgement numbers provide unique identifiers of the connection to the client and server, and provide confirmation that data was received. TCP flags are used to control the state of the TCP connection. Table 1 shows the TCP flags, two of which (ECE and CWR) were originally re-

IP Header

Version	Header length	Type of Service	Total length
Identification		Flags	Fragment offset
Time to Live	Protocol		Header checksum
Source address			
Destination address			
Options (optional)			

TCP Header

Source port			Destination port
Sequence Number			
Acknowledgement number			
Offset	Reserved	Flags	Window
Checksum			Urgent pointer
Options (optional)			

Figure 2: *IP and TCP header layouts.*

served (as shown in Figure 2), but are now being used to communicate congestion control capabilities as defined in RFC 3168 [23].

Table 1: *Summary of the TCP session bit flags*

Flag	Meaning
ECE[1]	ECN-Echo
CWR[1]	Congestion window reduced
URG	Urgent data
ACK	Valid acknowledgement
PSH	Push request
RST	Reset session
SYN	Synchronize sequence number
FIN	Final data

A typical TCP session for the transfer of data is shown in Figure 3. Before any data exchange takes place, the client and server must complete a three-way handshake. A sample handshake, captured by *tcpdump* [24], is shown below with the timestamp removed:

```
xx.xy.yz.1.48628 > xa.xb.xc.169.80: S 2914302952:2914302952(0) win 64240 <mss 1460> (DF)
xa.xb.xc.169.80 > xx.xy.yz.1.48628: S 407888030:407888030(0) ack 2914302953 win 32120 <mss 1460> (DF)
xx.xy.yz.1.48628 > xa.xb.xc.169.80: . ack 407888031 win 0
```

A connection exists when the handshake is complete, at which point data may be exchanged.

In summary, a TCP connection progresses as follows when the server initiates a clean tear-down [20,25,26]:

1. Client sends a SYN packet with sequence number J.
2. Server receives the SYN packet and sends its own SYN packet with a sequence number K. At the same time it acknowledges the client's SYN packet by sending an acknowledgement (ACK) packet with an acknowledgement number $J + 1$.
3. The client acknowledges the server's SYN packet by sending an ACK packet with a acknowledgement number $K + 1$. This establishes a full connection. The three-way handshake is complete and the client can then transmit data to the server.
4. The client and server exchange data, each sending an acknowledgement when data is received.

[1]Reserved bits.

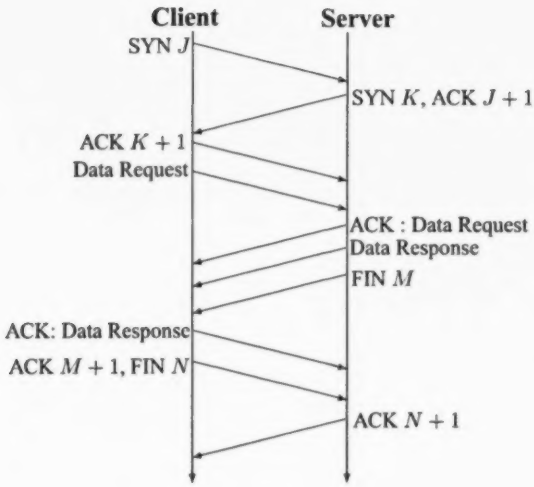

Figure 3: *TCP data packet exchange and data transfer.*

5. The server closes the connection by transmitting a FIN packet with sequence number M.

6. The client acknowledges the FIN packet with an ACK whose acknowledgement number is $M+1$(assuming no data transmitted). If D bytes of data are transmitted, then the acknowledgement number is $M+1+D$. The server may still send more data. The client closes the connection by transmitting its own FIN packet with sequence number N.

7. Finally the server transmits an ACK packet with sequence number $N+1$. That terminates the connection.

Either party may initiate termination of the connection, and the termination may not be as graceful as shown above. The connection may be terminated at any time by resetting it with a TCP RST packet.

2.2 Artificial Neural Networks

ANN models attempt to emulate the human brain through the dense interconnection of simple computational elements called neurons [27]. Each neuron is linked to some of its neighbours through synaptic connections of varying strengths. Learning is accomplished by continuously adjusting these connection strengths (weights)

until the overall network outputs the desired results. These weight adjustments are based on mathematical algorithms used in solving nonlinear optimization functions.

2.2.1 The Neuron

Similar to the biological nervous system, the basic computational element of an ANN is called the neuron or processing node. The neuron model is based on highly simplified considerations of the biological neuron. A simple node is shown in Figure 4, where N inputs are summed at the node. Each input u_i is connected to the

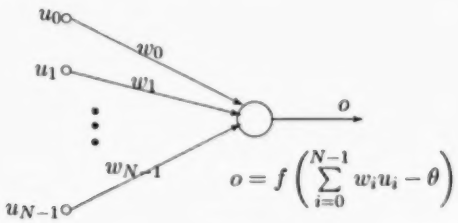

Figure 4: *The basic neuron*

processing node through the synaptic connections, which are represented by connection strengths called weights w_i. A bias term θ is also used at each node. The sum is fed through a transfer function f, called the activation function, to generate the output o. The signal flow is considered unidirectional as indicated by the arrows.

Although ANNs are constructed using this fundamental building block, there are significant differences in the architectures and driving fundamentals behind each ANN model.

2.2.2 The Activation Function

The activation function f plays a pivotal role in the functioning of the neuron. It determines the node output. As in Figure 4, the neuron output signal is given by:

$$o = f\left(\mathbf{w}^T \mathbf{u}\right) \tag{1}$$

where $\mathbf{w}$ is the weight vector defined as

$$\mathbf{w} \equiv \left[\begin{array}{cccc} w_1 & w_2 & \cdots & w_N \end{array} \right]^T$$

and the input vector u is defined as

$$\mathbf{u} \equiv \begin{bmatrix} u_1 & u_2 & \cdots & u_N \end{bmatrix}^T$$

There are many different types of activation functions f to choose from, depending on the application [2,27,28]. Some of the commonly used activation functions are shown in Figure 5. These activation functions are the *hard-limiter*, the *threshold logic*, and the *sigmoid*. Since real applications are usually modeled as continuous functions, the most commonly used continuous activation function is the sigmoid.

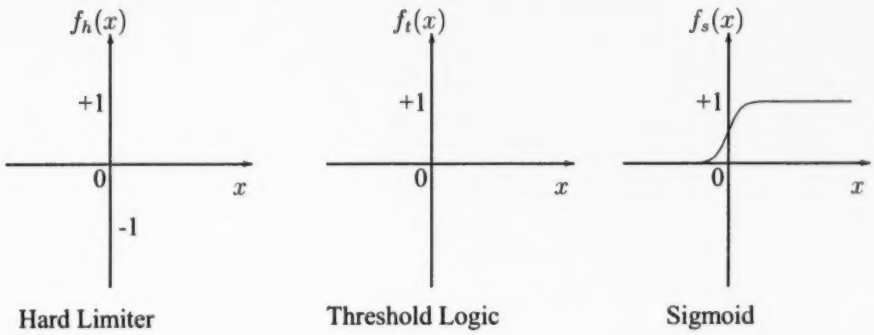

***Figure 5:** Activation functions*

Activation functions may be either unipolar, for positive output, or bipolar for output that may be positive or negative. For example, the bipolar sigmoidal activation function is defined as:

$$f(x) \equiv \frac{2}{1 + \exp^{-\lambda x}} - 1 \tag{2}$$

and the unipolar sigmoidal activation function is defined as

$$f(x) \equiv \frac{1}{1 + \exp^{-\lambda x}} \tag{3}$$

where λ is a constant.

A special case of an ANN is a single node based on the neuron model shown in Figure 4 and is called a *perceptron* after the work of Rosenblatt [27]. A perceptron consists of one or more neurons. If a continuous activation function is used, the neuron model is known as a *continuous perceptron*. A continuous perceptron is capable of classifying *linearly separable* classes of data of the form $ax+b$. Multiple nodes in this format form a single layer multi-node ANN capable of classifying linearly separable data patterns.

2.2.3 Multi-Layer ANNs

To emulate massively interconnected biological systems, ANNs have to be similarly interconnected. ANNs are the simple clustering of primitive artificial neurons. This clustering occurs by creating layers of neurons which are connected to one another. Figure 6 shows a multi-layer perceptron. An input layer interfaces with the outside world to receive inputs and an output layer provides the outside world with the network's outputs. The rest of the neurons are hidden from view, and are called *hidden layers*.

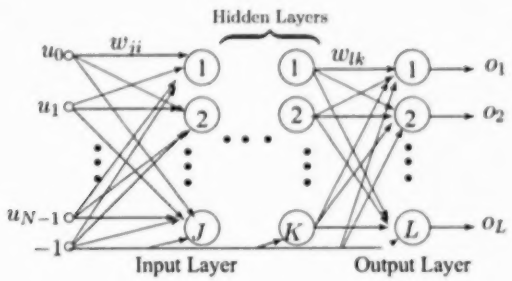

***Figure 6:** Multi-layer perceptron*

The objective of using a multi-layer perceptron is to be able to classify patterns that linear classifiers (single layer ANNs) are incapable of classifying. The most important attribute of multi-layer ANNs is that they can learn to classify a problem of any complexity. The biggest challenge is usually in deciding the number of hidden layers in an ANN.

Zurada [2] gives an extensive discussion on the design of the number and size of hidden layers in a given architecture; nevertheless, trial and error methods have been widely used. If the number of hidden layers is too large, the ANN architecture will have problems generalizing; it will simply memorize the training set, making it useless for use with new data sets.

Inter-layer connections within an ANN architecture can take the following forms [2]:

- In a *fully-connected* ANN, each neuron on one layer is connected to every neuron on the next layer.
- In a *partially-connected* ANN, a neuron on one layer does not have to be connected to all neurons on the next layer.

If signal flow direction is taken into consideration, these two architectures can be further refined:

- In a *feedforward* ANN, the neurons on one layer send their output to the neurons in the next layer, but they do not receive any input back from the neurons in the next layer.

- In a *bi-directional* ANN, the neurons on one layer may send their output to the next layer or the preceding layer, and the subsequent layers may also do the same.

- In a *hierarchical* ANN connection, the neurons of a lower layer may only communicate with neurons on the next level of layers.

- In a *resonance-connected* ANN, the layers have bi-directional connections, and they can continue sending messages across the connections a number of times until previously defined conditions are achieved.

In more sophisticated ANN structures the neurons communicate among themselves within a layer, this is known as intra-layer connections. These take the following two forms:

- In fully- or partially-connected *recurrent* networks, neurons within a layer communicate their outputs to neurons within the layer. This is done a number of times before they are allowed to send their outputs on to another layer.

- In *on-center/off-surround* ANNs, a neuron within a layer has excitatory connections to itself and its neighbors, and has inhibitory connections to other neurons. The neurons exchange their output signals a number of times until a winner is found. The winner is allowed to update its and its members' weights.

The overall architecture of an ANN depends on the mappings required, the type of input patterns, and the learning rules to be used.

2.2.4 ANN Training

Similar to the brain, ANNs learn from experience by changing the ANN's connection weights until a solution is found. The learning ability of an ANN is determined by its architecture and by the algorithm chosen for training. The training methods [27] fall into broad categories:

- In *unsupervised training*, hidden neurons find an optimum operating point by themselves, without external influence.

- *Supervised training* requires that the network be given sample input and output patterns to learn. It is guided through the learning process until a satisfactory optimum operating point or a predefined threshold is reached. The most common training termination criteria is by setting a training threshold.

Backpropagation training is a form of supervised learning that has proven highly successful in training multi-layered ANNs. Information about errors is filtered back through the system and is used to adjust the connections between the layers, thus improving performance.

ANNs can be trained *on-line* or *off-line*. In off-line training algorithms, its weights do not change after the successful completion of the initial training. This is the most common training approach; especially in supervised training. In on-line or real time learning, weights continuously change when the system is in operation [2].

2.2.5 Training Rules

There is a wide variety of learning rules that are used with ANNs. Error minimization algorithms are used to determine the convergence levels when updating weights. In general, all ANN learning involves the iterative updating of the connection weights until the desired convergence is achieved. Most training algorithms start by initializing the weights to 0 or very small random numbers. This weight update is given by:

$$\mathbf{w}^{k+1} = \mathbf{w}^k - \Delta\mathbf{w}^k \tag{4}$$

Equation 4 is the ANN *general learning rule* [2]. The numerous learning rules, which are variations of this rule, only differ by the mathematical algorithms used to update the connection weights, or more specifically to calculate the value of $\Delta\mathbf{w}^k$ at each iteration k. Some of the common training rules are as follows:

- In the *Hebbian* rule [2,28], the connection weight update $\Delta\mathbf{w}^k$ is proportional to the neuron's output. This was the first ANN learning rule [27,29].
- The *perceptron* rule [27] updates the weights based on the difference between the desired output d and the actual neuron's response o.
- The *delta* learning rule [27,29] is based on the minimisation of the mean square error (MSE) as represented by the error function E, as shown in Equation 5.

  $$\mathbf{w}^{k+1} = \mathbf{w}^k - \eta\nabla E\left(\mathbf{w}^k\right) \tag{5}$$

  where η is a learning constant, and ∇E is the gradient of the error function E, defined by:

  $$E_k = \frac{1}{2}\left(d^k - o^k\right)^2 \tag{6}$$

The objective is to iterate Equation 5 until the error E approaches zero (or a preset threshold value).

For an ANN with P training patterns, and K outputs, the *root-mean square error* (also known as the MSE [2]) is defined as:

$$E_{rms} = \frac{1}{PK}\sqrt{\sum_{p=1}^{P}\sum_{k=1}^{K}(d_{pk} - o_{pk})^2} \tag{7}$$

- The *Widrow-Hoff* [2,28] learning rule (sometimes called the *Least Mean Square* learning rule) is considered a special case of the delta learning rule in that the neuron output o is independent of the activation function f.

- The most widely used supervised training approach which is derived from the Widrow-Hoff algorithm is the *error backpropagation training algorithm*. As the name implies, the error $\Delta \mathbf{w}^k$ is propagated back into the previous layers. This is done one layer at a time, until the first layer is reached.

  Consider an ANN with one hidden layer, K outputs, J hidden nodes, I inputs, and P training patterns. The output layer weights are adjusted as follows:

$$w_{kj} = w_{kj} + \eta\delta_{ok}y_j, \quad \text{for k} = 1, \cdots, \text{K, j} = 1, \cdots, \text{J} \tag{8}$$

  where η is a learning constant and the output error $\delta_{\mathbf{ok}}$ is given by

$$\delta_{ok} = \frac{1}{2}(d_k - o_k)(1 - o_k^2), \quad \text{for k} = 1, 2, \cdots, \text{K} \tag{9}$$

  The weight update for the hidden layer is as follows:

$$w_{ji} = w_{ji} + \eta\delta_{yj}u_i, \quad \text{for k} = 1, \cdots, \text{K, i} = 1, \cdots, \text{I} \tag{10}$$

  where the output error $\delta_{\mathbf{yj}}$ is given by

$$\delta_{yj} = \frac{1}{2}(1 - y_j^2)\sum_{k=1}^{K}\delta_{ok}w_{kj}, \quad \text{for j} = 1, 2, \cdots, \text{J} \tag{11}$$

  The process is iteratively repeated until a preset threshold of the MSE (Equation 7) is achieved.

3 Packet Anomaly Detection Approach

In Section 3.1 we explore those TCP packet attributes that would enable an ANN classifier to identify normal and abnormal activity on a packet-by-packet basis. From these attributes, we form classifiers to be applied in the ANN in Section 3.2. In Section 3.3, we define the ANN model to be applied to this problem.

3.1 Attributes

A TCP packet can be characterised by the fields of the IP and TCP headers, shown in Figure 2. Some of the header field values are more indicative of an attack and misuse than others. In this section, we take a detailed look at how some of the attributes can be used to classify a packet as being suspicious. We also look at ways to combine these attributes in a way that would enhance our ability to detect a network anomaly by just looking at individual packets.

3.1.1 IP Addresses

Four byte IP addresses identify the source and destination of a packet. These attributes are used to identify suspicious IP addresses. Suspicious IP addresses are classified as follows:

- IP source address is the same as IP destination address (*land* DoS attack).
- IP address is a member of the private Internet address ranges, defined as 10.0.0.0/8, 172.16.0.0/12, and 192.168.0.0/16. As well, the loopback address 127.0.0.1 can indicate misuse.
- Source IP address is a broadcast or multicast address. TCP packets cannot be broadcast or multicast since a three way handshake is needed for communication.

This attribute can also be used to identify activity from "rogue" IP addresses associated with previously-identified suspicious activities, or from prohibited IP domains such as well-known AOL Instant Messenger (AIM) servers.

3.1.2 TCP Ports

TCP requires the use of ports to make a connection. When a client initiates a connection with a server, the client generally uses an *ephemeral* (*i.e.* short-lived) port and the server generally uses a *well-known* port, however any service may technically be run on any port [20, 22]. The Internet Assigned Names Authority (IANA) defines the following port ranges [30]:

Well-known:	ports 0–1023
Registered:	ports 1024–49151
Dynamic and/or private:	ports 49152–65535

The *ephemeral* ports were at one time defined as ranging from 1024 to 5000, but have since changed to use the *dynamic* port range as defined above.

The inclusion of the port attribute will reflect that a *well-known* port and an *ephemeral* port are almost always used in the TCP communications. Due to the variety of implementations of TCP, this model will generalize the *ephemeral* ports as the range 1024–65535. Some types of port scans may be detected through this attribute, as well as trojans and distributed denial of service (DDoS) activity.

This classifier will alert on packets with the following attributes:

- Either source or destination port of 0
- Both ports greater than 1023
- Both ports less than or equal to 1023

Port numbers and session flags form a set of aggregate attribute as discussed in Section 3.1.6

3.1.3 TCP Sequence and Acknowledgement Numbers

TCP uses 32-bit sequence numbers to order the data received. The sequence numbers consist of the initial sequence number (ISN), which represents the session establishment and an acknowledgement number [20]. In this model, we check to see if the sequence and acknowledgement numbers are valid. Sequence and acknowledgement numbers should be non-zero positive integers. Sequence and acknowledgement numbers together form an aggregate attribute, discussed in Section 3.1.6.

3.1.4 TCP Session Flags

The TCP flags of a given packet convey a message to the recipient. The valid combinations of the TCP flags listed in Table 1 and their meanings are shown in Table 2 [31]. Anything outside these combinations are viewed as suspicious. In addition, any use of the reserved bits [22] or a packet with no flags set (null session) are also viewed with suspicion. These are commonly used by attackers in OS fingerprinting. However, as per RFC 3168 [23] the reserved bits have now been proposed for use in TCP congestion control. Since explicit congestion control (ECN) is not yet an adopted standard (not all routers or network nodes implement it), alerts related to these bits should be treated with caution.

Table 2: *Valid TCP flag-byte combinations. The listed flag bits are set.*

Flag Combinations	*Function*
SYN	Request connection
SYN/ACK	Agree to open connection
ACK, PSH/ACK, ACK/URG, PSH/ACK/URG	Acknowledge receipt
FIN/ACK, FIN/PSH/ACK, FIN/PSH/ACK/URG	Request to close connection
RST, RST/ACK, RST/PSH, RST/URG, RST/PSH/ACK, RST/PSH/URG, RST/ACK/URG, RST/PSH/ACK/URG	Sever connection

3.1.5 Payload Size

The payload size of a TCP packet can be combined with other attributes to indicate anomalous activity, as listed in Section 3.1.6.

3.1.6 Composite Attributes

Other classifiers that identify anomalous activity must be built as a combination of the attributes described above. The following composite attributes are incorporated into the model:

- The combination of only a SYN flag bit set and non-zero payload.
- The combination of only a SYN flag bit set on a packet with a source port that is a *well-known* port number. This is a special case of the port attribute where we can include directionality.
- The combination of RST and ACK bits set and non-zero payload. Note, however, that some TCP implementations will attach a message stating the reason that a connection was torn down.
- Acknowledgement and sequence numbers equal to zero and flags RST or RST/ACK not set.

3.2 Classifiers

For maximum efficiency of the model, it is beneficial to minimize the total number of classifiers. While the attributes listed in Section 3.1 could each be applied individually, resulting in 10 classifiers, it is preferable to group attributes. For example,

the TCP session flags may be grouped to train a single classifier. In this work, however, it was established that instead of having classifiers for the individual set of attributes, we can combine the session flags with the payload size, and port number attributes.

In Table 3, the four classifiers that form a complete set for detection of TCP anomalies are shown. The attributes associated with each classifier are listed along with the number of inputs to each ANN classifier. Note that the fourth classifier contains the source port attribute, but is not used. This is included to demonstrate the case where one might want to apply a policy denying a certain service from a particular IP address or range of addresses.

Table 3: *The ANN classifier configurations. SEQ is the TCP sequence number and ACK is the TCP acknowledgement number.*

Classifier	Classifier Input Variables	Anomalies
Flags	TCP flags TCP ports Payload size	Bad flag combinations SYN with payload RST or RST/ACK with payload SYN with well-known source port
Ports	TCP ports	Both ports $>$ 1023 Both ports $\leq$ 1023 Source or destination port 0
Sequence	TCP sequence number TCP acknowledgement number TCP flags (2 inputs)	SEQ=ACK=0 and not R or RA
IP	IP addresses	Private IP addresses Source IP same as destination IP Broadcast IP addresses

The classifiers are modelled in this work such that attack data may trigger an alert in more than one classifier. However, if an anomalous packet raises an alert in one classifier, it does not necessarily mean that it will raise an alert in all the other classifiers. In some cases, there was more than one alert in a single packet. This would therefore trigger alerts from more than one classifier.

3.3 The ANN Packet Anomaly Detection Model

Because the network data we were dealing with in this work formed multiple ANN inputs and cannot be classified using a linear classifier (single-layer perceptron),

three-layer fully-connected feedforward ANNs were used. A hidden layer was required due to the non-linear mapping between input and output.

Each classifier is a three-layer feedforward perceptron (including a hidden layer). The number of nodes in each layer depends on the number of input variables to the classifier as listed in Table 4. With over 100 000 training samples at a time, the number of input nodes was initially selected to be $N+1$ [2,28,32] and the number of hidden layer nodes was initially selected to be $2N+1$, but less that $3N$ [27,33], where N is the number of inputs for the classifier. Starting with $N+1$ input nodes, the number of hidden layer nodes were adjusted until convergence was achieved.

Table 4: *The number of inputs and number of nodes in the layers of the ANN classifiers. There is one output node for all classifiers.*

Classifier	Inputs	Input nodes	Hidden layer nodes
Flags	10	11	25
Ports	2	3	7
Sequence	4	5	11
IP	3	4	9

The ANN model uses a unipolar sigmoidal activation function for the output node. This is because the classifier outputs (Section 3.2) cannot take on negative values. The sigmoidal activation function given in Equation 3 is used with λ=1.

Off-line supervised training was implemented with the error backpropagation training algorithm. By using off-line training, we avoid allowing an attacker to re-train the system. Error backpropagation training was used due to its success in other related work [12].

4 The Procedure

As discussed in Section 2.2, the ANN architecture is dependent on the attribute space under consideration. In order to be able to identify and classify different attacks and violations, this system employed a number of ANN classifiers. Each classifier was trained to detect and classify one or more attacks. In Figure 7, the stages in the implementation of the model are summarised.

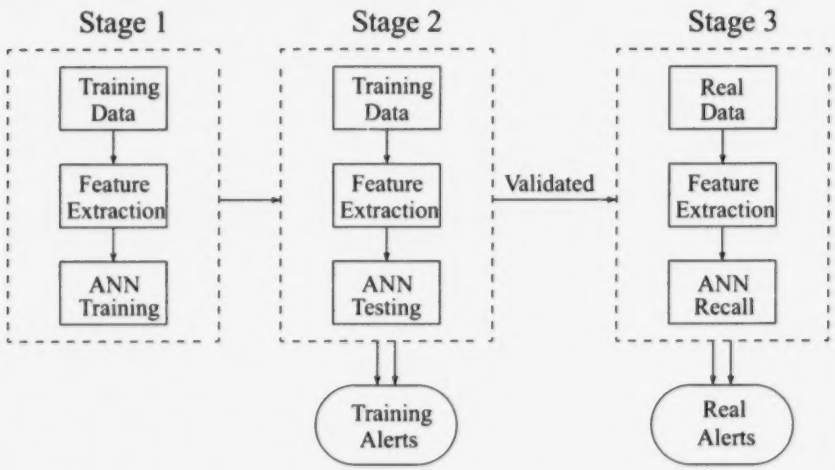

***Figure 7:** Implementation of the model.*

The first stage of the implementation of the model involved training the ANN. A suitable set of network data was obtained (in this case from the 1999 DARPA IDS Evaluation data set [19]) and the packet features (attributes) relevant to the classifiers were extracted from the data and stored in memory. The multi-classifier ANN was trained based on this data, which consisted of clean (attack free) traffic merged with traffic known to contain the attribute violations. In the second stage, the ANN was tested to verify that it had learned correctly. The alerts that came out of this stage should match the traffic in the training data that violated the attributes. If the ANN had been validated to have learned correctly, the ANN was applied to unknown, real data in the third stage.

4.1 ANN Training

The ANN was trained using *week 1* of the 1999 DARPA IDS Evaluation data set [19], which consists of five days of attack-free traffic collected in *tcpdump* format on a simulated network. The data collected on the internal sniffer was used.

Twenty-five percent of this data was manually modified to reflect the anomalous attribute definitions given in Section 3.1.

Training was achieved by making repeated presentations of the training data to the neural network. Weights were initialized to small random numbers [2,28]. Network training parameters (*i.e.*λ, error gradient, and learning rate η) were changed by trial and error methods whenever necessary to ensure that convergence was achieved.

Each classifier was trained with the same data sample for consistency: in a real-time implementation, all classifiers see each packet simultaneously. The training of each classifier was independent of the others, and the training results from one classifier did not affect the training of the next classifier. Data was presented to each ANN classifier until the individual convergence criteria was met (Equation 7). A set of 100 000 packets was presented 50 times to the ANN, and this process was repeated with subsequent sets of 100 000 packets until the MSE of 1×10^{-4} was achieved.

The ANN model was applied to the 1999 DARPA IDS evaluation data set [19] using the data collected by the *inside* sniffer, locke.eyrie.af.mil. The implementation was carried out using MATLAB release 16 with the aid of the Neural Network Toolbox [28]. The Network Traffic Analysis (NTA) toolbox [34] was also used to load and manipulate the network traffic.

4.2 ANN Testing and Recall

After the training of the multi-layer perceptrons, they were tested using the training dataset. The training dataset is only used to validate the performance of the ANN after the training. If the training was successful, the neural network's output should agree with the training dataset's expected output. If the training was not successful, it had to be done again, this time with different training parameters and different initial weights.

Once an ANN is successfully trained, it can be recalled using any network data available. In this case, the DARPA 1999 IDS Evaluation *week 5* data was used during recall. This data consists of 5 separate data sets, each with over 1×10^6 packets of TCP/IP traffic and a variety of documented attacks.

5 Results

5.1 Model Training and Validation

A sample training session curve for the first classifier (the IP classifier) is shown in Figure 8. Note that the IP classifier had originally been trained to detect all private IP addresses. Since the DARPA simulated network uses two private IP address domains, this classifier was easily modified and retrained.

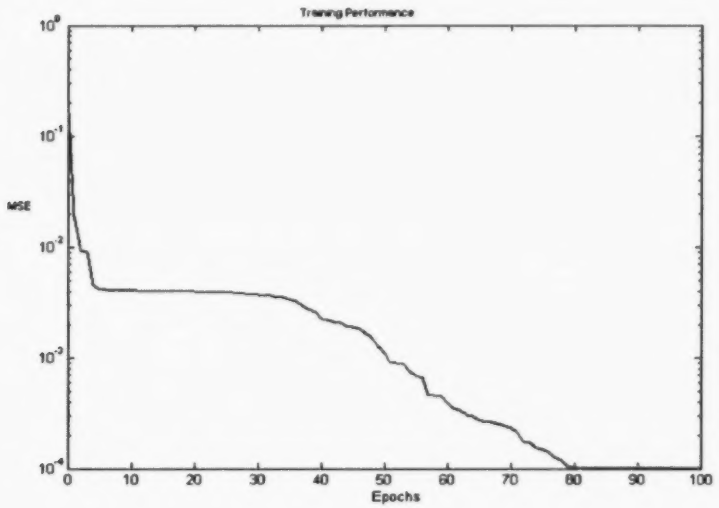

Figure 8: *The training mean square error.*

The target MSE (Equation 7) of 1×10^{-4} was met in 100 epochs of training; i.e. after two presentations. An epoch represents a complete pass through the ANN of 100 000 packets of training data. Each classifier had its own training parameters and convergence criteria. While the accelerated batch steepest descent training algorithm with an initial learning rate $\eta = 0.7$ and MSE of 1×10^{-4} was used in all classifiers during the final phase of the training, the minimum gradient for the four classifiers were respectively 1×10^{-12}, 1×10^{-6}, 1×10^{-10}, and 1×10^{-10}. In all classifiers, the number of epochs was initially set to 50 to allow successive training of different successive sets of training data (our application could only load 100 000 packets at a time).

The models were first tested by performing an ANN recall on the training data. This is an essential validation step for ANN training. Each fully trained classifier was presented with the training data, and the results were compared with the expected

output. In all cases, the validation was successful.

In the next section, we present the summarised alerts detected by the ANN classifiers. The detailed classifier alerts are presented in Annexes B to E.

5.2 Classifier Detection Performance for DARPA 1999 Data

The DARPA IDS Evaluation data from *week 5* was used to test the model. In this data, there is a limited number of attacks that can be found by examining the headers of single TCP packets. The majority of the documented attacks required an analysis of content, which is not included in this model. Attacks such as the portsweeps on 04/07/99 and 04/08/99, which use legitimate IP addresses, TCP port numbers, flags, and sequence/acknowledgement numbers were not detected by any classifier. This type of activity requires the examination of a series of packets for detection. Other documented attacks were not present in the data collected by the internal sniffer and therefore could not be detected. Table 5 shows the documented attacks that were detected by the model.

Modification of the IP classifier would allow one to detect activity based on access rules. For example, the *xlock* and *xsnoop* attacks could be detected via a classifier that has been configured to reflect the security policy of the network under surveillance.

The *dosnuke* attack may be detected if one modified the *Flags* classifier to alert on packets with the URG flags bit set. This, however, could also result in an increase in false positives as the URG flag is not uncommon.

5.2.1 False Positives

There were false positives related to our initial rule-set definitions and errors on the model's (and the NTA toolbox's) inability to handle fragmented packets. There were 1 076 false positives related to the "SYN with low source port" aggregate attribute in the *Flags* classifier. FTP data transfers generally appear as a SYN with source port 20, hence all FTP data transfers in the *week 5* data triggered an alert. Also, in the *week 5* data, the printer was configured to communicate using ports 515 and 1023, which triggered 50 alerts for the *Flags* classifier (SYN with low source port) and 609 alerts for the *Ports* classifier (low port to low port). SSH sessions produced 7 244 alerts for the same reasons.

Fragmentation in the *week 5* data was a source of 368 false positives in the *Flags* generated alerts. There were repeated telnet sessions where the client fragmented

Table 5: *Summary of DARPA 1999* week 5 *detects.*

Date	Attack	Classifier	Details
04/05/99	Portsweep	Flags Sequence #	Lone FIN packets SEQ=ACK=0
04/05/99	Neptune DoS	IP Flags Ports	Private IP address 10.20.30.40 SYN packets with low source ports Low ports to low ports
04/06/99	Ftpwrite	Ports	513–1023 Low port to low port (final stage of attack)
04/06/99	Neptune DoS	IP Flags Ports	Private IP address 10.20.30.40 SYN packets with low source ports Low ports to low ports
04/06/99	HTTP Tunnel	Ports	8000–32890 High port to high port
04/06/99	QueSO	Flags	Bad flag combinations
04/07/99	Netbus	Ports	1290–12345 High port to high port (final stage of attack)
04/07/99	QueSO	Flags	Bad flag combinations, responses detected
04/07/99	Portsweep	Flags Sequence #	Lone FIN packets SEQ=ACK=0
04/07/99	QueSO	Flags	Bad flag combinations
04/08/99	NTinfoscan	Ports Flags	Low port to low port SYN packets with low source ports
04/08/99	HTTP Tunnel	Ports	8000–32939 High port to high port
04/08/99	Satan	Flags Ports	SYN packets with low source ports High ports to high ports
04/08/99	NTinfoscan	Ports Flags	Low port to low port SYN packets with low source ports
04/09/99	Land DoS	IP Flags Ports	Source IP address same as destination IP address SYN packet with low source port Low port to low port
04/09/99	Portsweep	Flags Sequence #	Lone FIN packets SEQ=ACK=0
04/09/99	Neptune DoS	Flags Ports	SYN packets with low source ports Low ports to low ports

***Table 6:** Misclassifications detected by this approach.*

Classifier	False Positives	False Negatives	# Detected
Flags	1 444	0	2 383
IP	0	0	128 134
Sequence	0	0	1 032
Port	7 853	0	41 557
Total	9 297	0	173 106

every packet. The first fragment ended 4 bytes into the TCP header, giving only the source and destination port in the first packet. All other information, including TCP flags, was carried by subsequent packets. It was decided that packet reassembly would not be implemented as part of this model, and all such alerts were ignored.

Table 6 summarises the statistics for the model's false positives. Overall, out of 10 817 196 packets analysed, a total of 173 106 were detected as suspicious by the individual classifiers. The sequence numbers classifier had the best performance, with no false positives. The flags classifier was the worst affected by false positives which accounted for 60% of all the classifier's detects. However, when we factored in the original alert definitions, there are only 368 false positives for this classifier. The implementation of this classifier did not include the handling of fragmentation.

Except for the labeled attacks, mentioned at the beginning of this section, which our approach could not detect, there were no false negatives detected by any of the classifiers.

5.2.2 Additional DARPA Data Findings

The ANN produced some alerts that were not documented as an attack in the DARPA 1999 data. An undocumented *land* attack was found on 04/05/99. Attempts to access ports 8000 (previously used for the *HTTPtunnel* attack) and 9000 were detected on 04/05/99, 04/06/99 and 04/07/99.

Six HTTP sessions were detected on 04/07/99 and 04/08/99 where the client sends data 1 byte at a time. These were identified through an anomalous lone FIN packet sent at the start of each session. Further investigation revealed the anomalous data transfer behaviour. While the content of the packets was not found to be malicious, the behaviour is certainly suspicious and consumes bandwidth unnecessarily.

The *tcpreset* attack was expected to have been detected by the ANN, however no evidence of the attack was found in the ANN alerts. A manual inspection of the data also showed no evidence of the attack, so we conclude that the failure is not with the ANN classifiers. There were two cases where TCP reset packets were detected

by the *Ports* classifier, with source and destination port 123. These are odd, as all previous communication on these ports were UDP. Had this been an attempted *tcpreset* attack, these packets should have no effect.

6 Concluding Remarks

In this work, we have demonstrated the design and successful implementation of a system of multi-layer perceptron classifiers to detect suspicious TCP traffic at a packet-by-packet level. The ANN is capable of detecting attacks ranging from denial of service attacks and probing activity to activities violating a network security policy through access of prohibited IP domains (*e.g.* email, audio or video data transfers).

The alerts generated by the ANN approach were compared to the documented attacks in the DARPA 1999 IDS Evaluation data. The method was successful in detecting those documented attacks that are not dependent on detection of content or on analysis of a series of packets. Some of the attacks were discovered by performing an analysis of the other, legitimate, packets related to the alert. The classifiers can be re-trained easily to define access rules, which would allow for the detection of attacks that otherwise do not trigger an alert from our classifiers.

The DARPA data contained attacks and suspicious activity that was not documented. A *land* attack was discovered, as well as excessively fragmented telnet sessions and HTTP sessions where the client sends 1 byte of data per packet. While the latter two are not necessarily attacks, they are certainly cause for concern.

While there were no false negatives reported, the classifiers generated some false positives related to the original rule-set definitions and the model's handling of packet fragmentation. The classifiers read the subsequent packet fragments which don't contain the TCP header information, as full packets. As a result, the information read into the classifiers was incorrect, resulting in the false positives. The classifiers can be tailored to the network under surveillance to minimize these events as much as possible, but it would be difficult to eliminate them altogether.

This model also addresses one of ANN's limitations, specifically its usual inability to report the exact cause of an alert. By pointing to which classifier triggered the alert, we enable the operator to identify the source of the alarm. Future work will research on the possibility of expanding the output of the individual classifiers so that it would be easier to identify the exact source of a given attack.

By definition this model does not detect attacks that do not utilize TCP or that require an analysis of a sequence of packets. Future work will investigate how this model, or a derivative of it, would perform when multiple packets forming a session and multiple sessions are considered. Given that many of the attacks in the DARPA attack data fall into this category, further investigations of ANN models to include coordinated attacks and multiple events are warranted.

One application that may take advantage of the processing speed of ANNs is to use an ANN model in conjunction with real-time operating systems on network devices or firewalls where the need for processing speeds and precision are critical. The ANN model may be made to be easily configurable from a user interface (UI) to allow changes in ANN definitions and to allow off-line training.

For the size and complexity of the problem we are dealing with in this work, using ANNs may be too powerful to have significant advantages over using IDS rules like Snort. However, for possible layer 2 implementation in real-time on TCP/IP communications equipment, this model would potentially be much better than IDS rules in that it is faster and requires less computational effort.

Given the ever-changing definitions in data communications, the use of an ANN with supervised training may not be the best option since frequent retraining would be required. Rather than pursuing this further, in an effort to perfect it, it would be wise to explore the use of unsupervised training techniques such as the autoassociator. Efforts are already under way to explore the usage of the autoassociator in other intrusion detection work such as network event correlation and DDoS detection.

References

1. Dondo, M. (2003). An Overview of Computational Intelligence Techniques in Intrusion Detection Systems. In *IASTED International Conference on Neural Networks and Computational Intelligence*, pp. 102–107. Cancun, Mexico.

2. Zurada, J.M. (1992). Introduction to Artificial Neural Systems, New York: West Publishing Company.

3. Lunt, T.F. (1993). A survey of intrusion detection techniques. *Computers & Security*, **4**(12), 405–418.

4. Bace, R. (2000). Intrusion Detection, Indianapolis, IN: Macmillan Technical Publishing.

5. Bace, R. and Mell, P. (2001). Intrusion Detection Systems. *NIST Special Publications*. Available on-line at `http://csrc.nist.gov/publications/nistpubs/800-31/sp800-31.pdf`.

6. Kemmerer, R.A. and Vigna, G. (2002). Intrusion Detection. *IEEE Computer-Special publication on Security and Privacy*, pp. 27–29. Special publication on Security and Privacy.

7. Planquart, J.-P. (2001). Application of Neural Networks to Intrusion Detection. *SANS Institute*. Available on-line at `http://www.sans.org/rr/intrusion/neural.php`.

8. Cannady, J. and Mahaffey, J. (1998). The Application of Neural Networks to Misuse Detection. In *1st International Workshop on the Recent Advances in Intrusion Detection*, Lecture Notes in Computer Science. Available on-line at `http://www.raid-symposium.org/raid98/Prog\_RAID98/Talks.html{\#}Cannady\_34`.

9. Girardin, L. (1999). An eye on network intruder-administrator shootouts. In *Proceedings of the 1st Workshop on Intrusion Detection and Network Monitoring (ID '99)*, 1, Santa Clara, CA. Available on-line at `http://www.ubilab.org/publications/index.html`.

10. Zhang, Z., Li, J., Manikopoulos, C. N., Jorgenson, J., and Ucles, J. (2001). HIDE: a Hierarchical Network Intrusion Detection System Using Statistical Preprocessing and Neural Network Classification. In *Proceedings of the 2001 IEEE Workshop on Information Assurance and Security*, pp. 85–90. West Point, NY. Available on-line at `http://www.itoc.usma.edu/Workshop/2001/Authors/Submitted\_Abstracts/paperT2A2(19).pdf`.

11. Lindqvist, U. and Porras, P.A. (2001). eXpert-BSM: A Host-based Intrusion Detection Solution for Sun Solaris. In *Proceedings of the 17th Annual Computer Security Applications Conference (ACSAC 2001)*, pp. 240–251. New Orleans, Louisiana: IEEE Computer Society. Available on-line at http://www.sdl.sri.com/papers/expertbsm-acsac01/.

12. Ghosh, A.K. and Schwartzbard, A. (1999). A Study in Using Neural Networks for Anomaly and Misuse Detection. In *Proceedings of the Eighth USENIX Security Symposium*. Available on-line at http://www.usenix.org/events/sec99/full_papers/ghosh/ghosh.pdf.

13. Pack, D.J., Streilein, W., Webster, S., and Cunningham, R. (2002). Detecting HTTP Tunneling Activities. In *Proceedings of the 2002 IEEE Workshop on Information Assurance*. Available on-line at http://www.ll.mit.edu/IST/pubs/Pack-IEEE2002.pdf.

14. Ripley, B.D. (1996). Pattern Recognition and Neural Networks, 1 ed. Cambridge: Cambridge University Press.

15. Dickerson, J.E., Juslin, J., Koukousoula, O., and Dickerson, J.A. (2001). Fuzzy Intrusion Detection. In *IFSA World Congress and 20th North American Fuzzy Information Processing Society (NAFIPS) International Conference*, Vol. 3, pp. 1506–1510. Vancouver, British Columbia. Available on-line at http://clue.eng.iastate.edu/\~julied/publications/NAFIPS2001{\%-\%}20Fuzzy{\%}20Intrusion{\%}20Detection{\%}20v6.pdf.

16. Crosbie, M. and Spafford, E.H. (1995). Applying Genetic Programming to Intrusion Detection. In Siegel, E. V. and Koza, J. R., (Eds.), *Working Notes for the AAAI Symposium on Genetic Programming*, pp. 1–8. MIT, Cambridge, MA, USA: AAAI. Available on-line at http://www.aaai.org/.

17. Thomson, K., Miller, G.J., and Wilder, R. (1997). Wide-area traffic patterns and characteristics. *IEEE Network*, **11**(6), 10–23.

18. Sedayao, J. (1995). World Wide Web network traffic patterns. In *40th IEEE Computer Society International Conference (COMPCON'95)*. Available on-line at http://csdl.computer.org/comp/proceedings/compcon/1995/7029/00/70290008abs.htm.

19. DARPA (1999). 1999 DARPA Intrusion Detection Evaluation Data Set Overview. *MIT: DARPA Intrusion Evaluation*. Available on-line at http://www.ll.mit.edu/IST/ideval/data/1999/1999_data_index.html.

20. Stevens, W.R. (1994). TCP/IP Illustrated : The Protocols, Vol. 1. Reading MA: Addison-Wesley.

21. Hornig, C. (1984). A Standard for the Transmission of IP Datagrams over Ethernet Networks: RFC894. Technical Report. Symbolics Cambridge Research Centre. Available on-line at http://ftp.rfc-editor.org/in-notes/rfc894.txt.

22. Northcutt, S. and Novak, J. (2000). Network Intrusion Detection : An Analyst's Handbook, 2 ed. Indianapolis, Indiana: New Riders.

23. Ramakrishnan, K., Floyd, S., and Black, D. (2001). The Addition of Explicit Congestion Notification (ECN) to IP: RFC 3168. Technical Report. Available on-line at http://ftp.rfc-editor.org/in-notes/rfc3168.txt.

24. The Tcpdump Group (Jan. 2002). Tcpdump (Online). http://www.tcpdump.org (28 Aug 2003).

25. DARPA Internet Program Protocol Specification (1981). Transmission Control Protocol: RFC793. Technical Report. DARPA. Available on-line at http://ftp.rfc-editor.org/in-notes/rfc793.txt.

26. Touch, J., Heidemann, J., and Obraczka, K. (1996). Analysis of HTTP Performance. Technical Report. USC/Information Sciences Institute. Available on-line at http://www.isi.edu/lsam/publications/http-perf.

27. Lippman, R.P. (1987). An Introduction to Computing with Neural Nets. In *IEEE ASSP Magazine*, pp. 4–22.

28. Demuth, H. and Beale, M. (2001). Neural Network Toolbox, MathWorks Version 4.

29. Widrow, B. and Lehr, M.A. (1990). 30 Years of Adaptive Neural Networks: Perceptron, Madaline, and Backpropagation. In *IEEE Proceedings*, Vol. 78, pp. 1415–1442.

30. The Internet Assigned Numbers Authority (IANA) (Apr. 2004). Port Numbers (Online). IANA. http://www.iana.org/assignments/port-numbers (19 Apr 2004).

31. Treurniet, J. and Lefebvre, J.H. (2003). A Finite State Machine Model of TCP Connections in the Transport Layer. (DRDC Ottawa TM 2003-139). Defence R&D Canada – Ottawa.

32. Lawrence, Steve, Giles, C. Lee, and Tsoi, A.C. (1997). Lessons in Neural Network Training: Overfitting May be Harder than Expected. In *Proceedings of the Fourteenth National Conference on Artificial Intelligence, AAAI-97*, pp. 540–545. Menlo Park, California: AAAI Press.

33. Rangsaneri, Yuttapong, Thitimajshima, Punya, and Promcharoen, Somying (1998). A Study of Neural Network Classification of Jers-1/Ops Images. In *Proceedings of the 1998 Asian Conference on Remote Sensing*. Available on-line at `http://www.gisdevelopment.net/aars/acrs/1998/ps1/ps1012.shtml`.

34. Gregoire, M. and Lefebvre, J.H. (2003). Network Traffic Analysis Toolbox. Technical Report. DRDC-Ottawa (In preparation).

Annex A
1999 DARPA Simulation Network

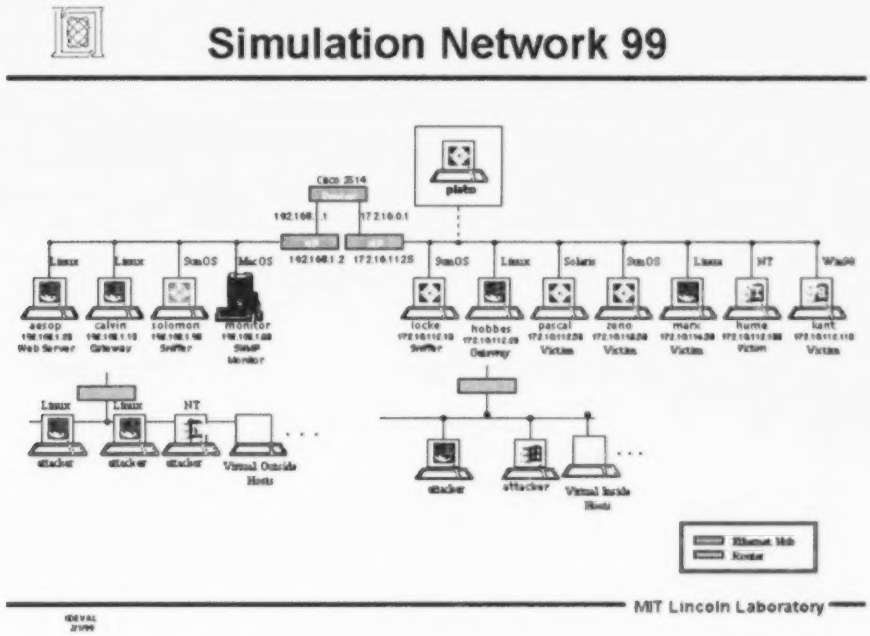

Figure A.1: *The 1999 DARPA simulation network.*

Annex B
Summary of *IP* Classifier Results

The *IP* classifier uses source and destination IP address and source port attributes to discover:

- Private IP addresses
- Source IP same as destination IP
- Broadcast IP addresses

With the use of the source port attribute, it can be trained to detect unauthorized services from known servers, however this was not applied.

```
Portscan, sequential dest ports on one host: Neptune DoS
1999-04-05 22:03:55.501770 10.20.30.40:4673 > 172.16.112.50:1: S 2187784450:2187784450(0) win 242
1999-04-05 22:03:55.502082 172.16.112.50:1 > 10.20.30.40:4673: R 0:0(0) ack 2187784451 win 0
...
1999-04-05 22:10:45.160028 10.20.30.40:4497 > 172.16.112.50:1024: S 2176250114:2176250114(0) win 242
1999-04-05 22:10:45.160201 172.16.112.50:1024 > 10.20.30.40:4497: R 0:0(0) ack 2176250115 win 0

Repeated, another host: Neptune DoS
1999-04-06 15:38:00.454642 10.20.30.40:4631 > 172.16.114.50:1: S 2185031938:2185031938(0) win 242
1999-04-06 15:38:00.454862 172.16.114.50:1 > 10.20.30.40:4631: R 0:0(0) ack 2185031939 win 0
...
1999-04-06 15:51:39.670267 10.20.30.40:4535 > 172.16.114.50:1024: S 2178740482:2178740482(0) win 242
1999-04-06 15:51:39.670566 172.16.114.50:1024 > 10.20.30.40:4535: R 0:0(0) ack 2178740483 win 0

Delayed SA responses to SYN packets to port 25 in the above DoS
1999-04-06 15:53:28.607993 172.16.114.50:25 > 10.20.30.40:54042: S 2843644623:2843644623(0) ack 1128329475 win 31744
...
1999-04-06 16:01:28.668890 172.16.114.50:25 > 10.20.30.40:56090: S 3917236053:3917236053(0) ack 1262547203 win 31744

Land attack DoS
1999-04-09 18:32:17.628397 172.16.113.50:25 > 172.16.113.50:25: S 3868:3868(0) win 2048
```

Annex C
Summary of *Flags* Classifier Results

The *Flags* classifier uses TCP flags, ports and payload size to discover:

- Bad flag combinations
- SYN with payload
- RST or RST/ACK with payload
- SYN with well-known source port

```
False positives: SYN from port 20 (ftp-data)

FIN scan: Lone FIN (bad flags) to random destination ports on one host
1999-04-05 13:43:08.073616 208.240.124.83:43170 > 172.16.112.50:3: F 0:0(0) win 2048
...
1999-04-05 13:46:50.927546 208.240.124.83:62309 > 172.16.112.50:9: F 0:0(0) win 2048

Undocumented: Land attack
1999-04-05 16:48:08.463617 172.16.112.50:25 > 172.16.112.50:25: S 3868:3868(0) win 2048

False positive: SYN from low port 1023 to printer port
1999-04-05 18:01:50.299745 172.16.113.50:1023 > 172.16.112.50:515: S 812224000:812224000(0) win 4096
1999-04-05 18:01:50.300261 172.16.112.50:515 > 172.16.113.50:1023: S 2939188589:2939188589(0) ack 812224001 win 8760
1999-04-05 19:21:36.605549 172.16.113.50:1023 > 172.16.112.50:515: S 1425152000:1425152000(0) win 4096
1999-04-05 19:21:36.606071 172.16.112.50:515 > 172.16.113.50:1023: S 3547480582:3547480582(0) ack 1425152001 win 8760
1999-04-05 20:14:23.621866 172.16.113.50:1023 > 172.16.112.50:515: S 1831296000:1831296000(0) win 4096
1999-04-05 20:14:23.622353 172.16.112.50:515 > 172.16.113.50:1023: S 3951426335:3951426335(0) ack 1831296001 win 8760
1999-04-05 20:19:18.412374 172.16.113.50:1023 > 172.16.112.50:515: S 1869312000:1869312000(0) win 4096
1999-04-05 20:19:18.412714 172.16.112.50:515 > 172.16.113.50:1023: S 3989044364:3989044364(0) ack 1869312001 win 8760

Neptune DoS: SYN from randomized low ports to random destination ports on one host
1999-04-05 22:04:00.320655 10.20.30.40:66 > 172.16.112.50:12: S 1885860098:1885860098(0) win 242
...

False positive: SYN from low port 1023 to SSH and printer ports
1999-04-06 13:19:18.178225 172.16.112.50:1023 > 172.16.112.20:22: S 751177833:751177833(0) win 8760
1999-04-06 13:19:18.178539 172.16.112.20:22 > 172.16.112.50:1023: S 657409615:657409615(0) ack 751177834 win 32736
1999-04-06 14:32:42.824842 172.16.114.207:1023 > 172.16.112.50:513: S 539785390:539785390(0) win 512
1999-04-06 14:32:42.825206 172.16.112.50:513 > 172.16.114.207:1023: S 322000618:322000618(0) ack 539785391 win 8760
1999-04-06 15:37:44.493847 172.16.113.50:1023 > 172.16.112.50:515: S 1812288001:1812288001(0) win 4096
1999-04-06 15:37:44.494172 172.16.112.50:515 > 172.16.113.50:1023: S 819369802:819369802(0) ack 1812288002 win 8760

Neptune DoS: SYN from randomized low ports on one host
1999-04-06 15:38:05.213565 10.20.30.40:24 > 172.16.114.50:6: S 1883107586:1883107586(0) win 242
...
1999-04-06 15:47:13.133721 10.20.30.40:899 > 172.16.114.50:691: S 1940451586:1940451586(0) win 242

False positive: SYN from low port 1023 to printer port
1999-04-06 15:47:13.248573 172.16.113.50:1023 > 172.16.112.50:515: S 1885376001:1885376001(0) win 4096
1999-04-06 15:47:13.249954 172.16.112.50:515 > 172.16.113.50:1023: S 891994416:891994416(0) ack 1885376002 win 8760

False positive: SYN from low port 1023 to printer port
1999-04-06 16:24:02.399909 172.16.113.50:1023 > 172.16.112.50:515: S 20672000:20672000(0) win 4096
1999-04-06 16:24:02.400289 172.16.112.50:515 > 172.16.113.50:1023: S 1173644809:1173644809(0) ack 20672001 win 8760
```

```
1999-04-06 17:44:03.086280 172.16.113.50:1023 > 172.16.112.50:515: S 635904000:635904000(0) win 4096
1999-04-06 17:44:03.086795 172.16.112.50:515 > 172.16.113.50:1023: S 1788027537:1788027537(0) ack 635904001 win 8760
1999-04-06 19:19:51.758595 172.16.113.50:1023 > 172.16.112.50:515: S 1372864000:1372864000(0) win 4096
1999-04-06 19:19:51.759129 172.16.112.50:515 > 172.16.113.50:1023: S 2517548376:2517548376(0) ack 1372864001 win 8760

QueSO: bad flags, no response detected
1999-04-06 20:54:14.044298 199.227.99.125:26873 > 172.16.113.50:25: S 1924662232:1924662232(0) ack 0 win 4660
1999-04-06 20:54:16.063980 199.227.99.125:26874 > 172.16.113.50:25: F 1924662232:1924662232(0) win 4660
1999-04-06 20:54:39.289140 199.227.99.125:26876 > 172.16.113.50:25: SF 1924662232:1924662232(0) win 4660
1999-04-06 20:54:41.308678 199.227.99.125:26877 > 172.16.113.50:25: P 1924662232:1924662232(0) win 4660
1999-04-06 20:55:02.514404 199.227.99.125:26878 > 172.16.113.50:25: SWE 1924662232:1924662232(0) win 4660

Undocumented: Anomalous HTTP session -- investigation shows 1 byte of data transferred per packet
1999-04-07 12:39:42.898774 206.48.44.50:2295 > 172.16.114.50:80: F 3208635203:3208635203(0) win 0
1999-04-07 13:15:28.885706 206.48.44.50:2297 > 172.16.114.50:80: F 4070161920:4070161920(0) win 0
1999-04-07 13:23:51.573858 206.48.44.50:2299 > 172.16.114.50:80: F 277741562:277741562(0) win 0

QueSO: bad flags, with response
1999-04-07 15:34:13.020863 197.182.91.233:16446 > 172.16.114.50:23: S 1150614957:1150614957(0) ack 0 win 4660
1999-04-07 15:36:14.014324 197.182.91.233:16447 > 172.16.114.50:23: F 1150614957:1150614957(0) win 4660
1999-04-07 15:40:35.192014 197.182.91.233:16449 > 172.16.114.50:23: SF 1150614957:1150614957(0) win 4660
1999-04-07 15:40:35.192357 172.16.114.50:23 > 197.182.91.233:16449: SF 4180280353:4180280353(0) ack 1150614958 win 31744
1999-04-07 15:42:55.373880 197.182.91.233:16450 > 172.16.114.50:23: P 1150614957:1150614957(0) win 4660
1999-04-07 15:45:15.555801 197.182.91.233:16451 > 172.16.114.50:23: SWE 1150614957:1150614957(0) win 4660
1999-04-07 15:45:15.556150 172.16.114.50:23 > 197.182.91.233:16451: SWE 2132661837:2132661837(0) ack 1150614958 win 3174

False positive: SYN from low port 1023 to printer port
1999-04-07 15:51:32.790147 172.16.113.50:1023 > 172.16.112.50:515: S 1968192001:1968192001(0) win 4096
1999-04-07 15:51:32.790651 172.16.112.50:515 > 172.16.113.50:1023: S 1921698939:1921698939(0) ack 1968192002 win 8760

FIN scan: lone FIN (bad flag)
1999-04-07 16:37:05.119686 204.97.153.43:33731 > 172.16.114.50:1: F 0:0(0) win 3072
1999-04-07 16:37:11.119509 204.97.153.43:33732 > 172.16.114.50:1: F 0:0(0) win 3072
1999-04-07 16:38:11.212840 204.97.153.43:48334 > 172.16.114.50:2: F 0:0(0) win 4096
1999-04-07 16:39:11.271440 204.97.153.43:36206 > 172.16.114.50:3: F 0:0(0) win 1024
1999-04-07 16:40:11.330299 204.97.153.43:34897 > 172.16.114.50:4: F 0:0(0) win 4096
1999-04-07 16:41:11.398469 204.97.153.43:44837 > 172.16.114.50:5: F 0:0(0) win 3072
1999-04-07 16:42:11.476742 204.97.153.43:57319 > 172.16.114.50:6: F 0:0(0) win 4096
1999-04-07 16:43:11.555454 204.97.153.43:42505 > 172.16.114.50:7: F 0:0(0) win 2048
1999-04-07 16:43:17.563378 204.97.153.43:42506 > 172.16.114.50:7: F 0:0(0) win 2048
1999-04-07 16:44:23.701950 204.97.153.43:47885 > 172.16.114.50:8: F 0:0(0) win 4096
1999-04-07 16:45:24.019973 204.97.153.43:47234 > 172.16.114.50:9: F 0:0(0) win 4096
1999-04-07 16:45:30.025698 204.97.153.43:47235 > 172.16.114.50:9: F 0:0(0) win 4096
1999-04-07 16:46:36.147226 204.97.153.43:53912 > 172.16.114.50:10: F 0:0(0) win 4096

False positive: SYN from low port 1021 to SSH
1999-04-07 16:48:18.389375 206.48.44.50:1021 > 172.16.114.50:22: S 3871009220:3871009220(0) win 512
1999-04-07 16:48:18.389717 172.16.114.50:22 > 206.48.44.50:1021: S 911077798:911077798(0) ack 3871009221 win 31744

QueSO: bad flags, no response
1999-04-07 17:43:16.262426 172.16.114.169:13697 > 172.16.112.50:25: S 552908031:552908031(0) ack 0 win 4660
1999-04-07 17:46:35.406694 172.16.114.169:13698 > 172.16.112.50:25: F 552908031:552908031(0) win 4660
1999-04-07 17:53:15.734876 172.16.114.169:13700 > 172.16.112.50:25: SF 552908031:552908031(0) win 4660
1999-04-07 17:56:19.733439 172.16.114.169:13701 > 172.16.112.50:25: P 552908031:552908031(0) win 4660
1999-04-07 17:59:23.732127 172.16.114.169:13702 > 172.16.112.50:25: SWE 552908031:552908031(0) win 4660

False positive: SYN from low port 1023 to printer port
1999-04-07 21:03:06.591166 172.16.113.50:1023 > 172.16.112.50:515: S 75840000:75840000(0) win 4096
1999-04-07 21:03:06.591482 172.16.112.50:515 > 172.16.113.50:1023: S 5138528:5138528(0) ack 75840001 win 8760

Undocumented: Anomalous HTTP session -- investigation shows 1 byte of data transferred per packet
1999-04-08 12:43:38.772729 206.48.44.50:3759 > 172.16.114.50:80: F 242486627:242486627(0) win 0
1999-04-08 12:43:46.552168 206.48.44.50:3821 > 172.16.114.50:80: F 2049880209:2049880209(0) win 0
1999-04-08 12:43:57.467964 206.48.44.50:3822 > 172.16.114.50:80: F 1554285451:1554285451(0) win 0
```

```
False positive: SYN from low port 1023 to printer port
1999-04-08 13:13:43.105379 172.16.113.50:1023 > 172.16.112.50:515: S 706176001:706176001(0) win 4096
1999-04-08 13:13:43.105721 172.16.112.50:515 > 172.16.113.50:1023: S 706994143:706994143(0) ack 706176002 win 8760
1999-04-08 13:18:28.613968 172.16.113.50:1023 > 172.16.112.50:515: S 742720001:742720001(0) win 4096
1999-04-08 13:18:28.614298 172.16.112.50:515 > 172.16.113.50:1023: S 743524233:743524233(0) ack 742720002 win 8760
1999-04-08 13:36:12.435355 172.16.113.50:1023 > 172.16.112.50:515: S 879232001:879232001(0) win 4096
1999-04-08 13:36:12.436765 172.16.112.50:515 > 172.16.113.50:1023: S 878153743:878153743(0) ack 879232002 win 8760

False positive: SYN from low port 1023 to SSH
1999-04-08 13:43:18.774060 206.48.44.50:1023 > 172.16.114.50:22: S 184755503:184755503(0) win 512
1999-04-08 13:43:18.776468 172.16.114.50:22 > 206.48.44.50:1023: S 889901208:889901208(0) ack 184755504 win 31744

SATAN: Lots of connections here, SYN from low port to port 111 (RPC)
1999-04-08 18:58:22.479377 209.74.60.168:878 > 172.16.114.50:111: S 2170743131:2170743131(0) win 512
...
1999-04-09 05:46:54.595297 207.136.86.223:941 > 172.16.115.87:111: S 3524467774:3524467774(0) win 512

False positive: SYN from low port 1023 to printer port
1999-04-09 13:16:44.668151 172.16.113.50:1023 > 172.16.112.50:515: S 770752001:770752001(0) win 4096
1999-04-09 13:16:44.668503 172.16.112.50:515 > 172.16.113.50:1023: S 772376795:772376795(0) ack 770752002 win 8760

False positive: SYN from low port 1023 to printer port
1999-04-09 15:30:54.969911 172.16.113.50:1023 > 172.16.112.50:515: S 1806144001:1806144001(0) win 4096
1999-04-09 15:30:54.970185 172.16.112.50:515 > 172.16.113.50:1023: S 1800094144:1800094144(0) ack 1806144002 win 8760
1999-04-09 15:33:13.269244 172.16.113.50:1023 > 172.16.112.50:515: S 1823872001:1823872001(0) win 4096
1999-04-09 15:33:13.269519 172.16.112.50:515 > 172.16.113.50:1023: S 1817483065:1817483065(0) ack 1823872002 win 8760
1999-04-09 15:42:55.529964 172.16.113.50:1023 > 172.16.112.50:515: S 1898624001:1898624001(0) win 4096
1999-04-09 15:42:55.531288 172.16.112.50:515 > 172.16.113.50:1023: S 1892983054:1892983054(0) ack 1898624002 win 8760

FIN scan: lone FIN (bad flags)
1999-04-09 15:52:06.231054 206.186.80.111:59543 > 172.16.113.50:79: F 0:0(0) win 3072
1999-04-09 15:52:12.256155 206.186.80.111:59544 > 172.16.113.50:79: F 0:0(0) win 3072
1999-04-09 15:53:58.359515 206.186.80.111:51887 > 172.16.113.50:7: F 0:0(0) win 3072
1999-04-09 15:54:04.372822 206.186.80.111:51888 > 172.16.113.50:7: F 0:0(0) win 3072
1999-04-09 15:55:50.468902 206.186.80.111:57112 > 172.16.113.50:9: F 0:0(0) win 2048
1999-04-09 15:55:56.489620 206.186.80.111:57113 > 172.16.113.50:9: F 0:0(0) win 2048
1999-04-09 15:57:42.595667 206.186.80.111:35145 > 172.16.113.50:19: F 0:0(0) win 1024
1999-04-09 15:57:48.616220 206.186.80.111:35146 > 172.16.113.50:19: F 0:0(0) win 1024

False positive: SYN from low port 1023 to printer port
1999-04-09 16:17:09.562055 172.16.113.50:1023 > 172.16.112.50:515: S 14016000:14016000(0) win 4096
1999-04-09 16:17:09.563467 172.16.112.50:515 > 172.16.113.50:1023: S 2154409720:2154409720(0) ack 14016001 win 8760
1999-04-09 16:27:22.135713 172.16.113.50:1023 > 172.16.112.50:515: S 92800000:92800000(0) win 4096
1999-04-09 16:27:22.136024 172.16.112.50:515 > 172.16.113.50:1023: S 2233128703:2233128703(0) ack 92800001 win 8760
1999-04-09 16:36:49.374774 172.16.113.50:1023 > 172.16.112.50:515: S 165440000:165440000(0) win 4096
1999-04-09 16:36:49.375085 172.16.112.50:515 > 172.16.113.50:1023: S 2304936803:2304936803(0) ack 165440001 win 8760
1999-04-09 18:21:51.699624 172.16.113.50:1023 > 172.16.112.50:515: S 973696000:973696000(0) win 4096
1999-04-09 18:21:51.699996 172.16.112.50:515 > 172.16.113.50:1023: S 3105260648:3105260648(0) ack 973696001 win 8760
1999-04-09 18:23:00.486114 172.16.113.50:1023 > 172.16.112.50:515: S 982592000:982592000(0) win 4096
1999-04-09 18:23:00.486446 172.16.112.50:515 > 172.16.113.50:1023: S 3114280893:3114280893(0) ack 982592001 win 8760

Land attack: SYN from low port
1999-04-09 18:32:17.628397 172.16.113.50:25 > 172.16.113.50:25: S 3868:3868(0) win 2048

Neptune DoS: SYN from randomized low ports to port 21
1999-04-09 22:29:56.680704 11.21.31.41:61 > 172.16.113.50:21: S 1885532418:1885532418(0) win 242
1999-04-09 22:29:56.700594 11.21.31.41:317 > 172.16.113.50:21: S 1902309634:1902309634(0) win 242
1999-04-09 22:29:56.720598 11.21.31.41:573 > 172.16.113.50:21: S 1919086850:1919086850(0) win 242
1999-04-09 22:29:56.740580 11.21.31.41:829 > 172.16.113.50:21: S 1935864066:1935864066(0) win 242
1999-04-09 22:30:01.849571 11.21.31.41:62 > 172.16.113.50:21: S 1885597954:1885597954(0) win 242
1999-04-09 22:30:01.869562 11.21.31.41:318 > 172.16.113.50:21: S 1902375170:1902375170(0) win 242
1999-04-09 22:30:01.889585 11.21.31.41:574 > 172.16.113.50:21: S 1919152386:1919152386(0) win 242
1999-04-09 22:30:01.909579 11.21.31.41:830 > 172.16.113.50:21: S 1935929602:1935929602(0) win 242
```

Annex D
Summary of *Ports* Classifier Results

The *Ports* classifier uses the TCP ports to discover:

- Both ports > 1023
- Both ports ≤ 1023
- Source or destination port 0

```
Undocumented: Repeated connection attempts to port 8000 (HTTPtunnel?)
1999-04-05 16:04:56.651007 172.16.112.50:32914 > 196.37.75.158:8000: S 2050946967:2050946967(0) win 8760
1999-04-05 16:04:56.651777 196.37.75.158:8000 > 172.16.112.50:32914: R 0:0(0) ack 2050946968 win 0

Undocumented: Land attack -- low port - low port
1999-04-05 16:48:08.463617 172.16.112.50:25 > 172.16.112.50:25: S 3868:3868(0) win 2048

Neptune DoS: Only the low-low and high-high port combinations are caught with this classifier
1999-04-05 22:04:00.320655 10.20.30.40:66 > 172.16.112.50:12: S 1885860098:1885860098(0) win 242
1999-04-05 22:04:00.320827 172.16.112.50:12 > 10.20.30.40:66: R 0:0(0) ack 1885860099 win 0
...
1999-04-05 22:10:45.160028 10.20.30.40:4497 > 172.16.112.50:1024: S 2176250114:2176250114(0) win 242
1999-04-05 22:10:45.160201 172.16.112.50:1024 > 10.20.30.40:4497: R 0:0(0) ack 2176250115 win 0

Possible Tcpreset attack: low port - low port
1999-04-06 12:12:31.599205 172.16.112.50:123 > 172.16.112.10:123: R 470091505:470091505(0) win 1
1999-04-06 12:13:35.594085 172.16.112.50:123 > 172.16.112.10:123: R 470091505:470091505(0) win 1
1999-04-06 12:14:39.577410 172.16.112.50:123 > 172.16.112.10:123: R 470091505:470091505(0) win 1
1999-04-06 12:15:43.561582 172.16.112.50:123 > 172.16.112.10:123: R 470091505:470091505(0) win 1
1999-04-06 12:16:47.555796 172.16.112.50:123 > 172.16.112.10:123: R 470091505:470091505(0) win 1
1999-04-06 12:17:51.539983 172.16.112.50:123 > 172.16.112.10:123: R 470091505:470091505(0) win 1
1999-04-06 12:18:55.524810 172.16.112.50:123 > 172.16.112.10:123: R 470091505:470091505(0) win 1
1999-04-06 12:19:59.508369 172.16.112.50:123 > 172.16.112.10:123: R 470091505:470091505(0) win 1

Ftpwrite: final stage of attack -- low port - low port
1999-04-06 14:32:42.824842 172.16.114.207:1023 > 172.16.112.50:513: S 539785390:539785390(0) win 512

Neptune DoS: low port - low port and high port - high port detected
1999-04-06 15:38:05.213565 10.20.30.40:24 > 172.16.114.50:6: S 1883107586:1883107586(0) win 242
1999-04-06 15:38:05.213760 172.16.114.50:6 > 10.20.30.40:24: R 0:0(0) ack 1883107587 win 0
...
1999-04-06 15:51:39.670267 10.20.30.40:4535 > 172.16.114.50:1024: S 2178740482:2178740482(0) win 242
1999-04-06 15:51:39.670566 172.16.114.50:1024 > 10.20.30.40:4535: R 0:0(0) ack 2178740483 win 0

HTTPtunnel: high port - high port
1999-04-06 16:04:57.810365 172.16.112.50:32890 > 196.37.75.158:8000: S 1026608069:1026608069(0) win 8760
1999-04-06 16:04:57.811115 196.37.75.158:8000 > 172.16.112.50:32890: R 0:0(0) ack 1026608070 win 0
1999-04-06 16:05:27.807153 172.16.112.50:32891 > 196.37.75.158:8000: S 1030515823:1030515823(0) win 8760
1999-04-06 16:05:27.807983 196.37.75.158:8000 > 172.16.112.50:32891: R 0:0(0) ack 1030515824 win 0
1999-04-06 16:05:57.804357 172.16.112.50:32892 > 196.37.75.158:8000: S 1034302995:1034302995(0) win 8760
1999-04-06 16:05:57.805050 196.37.75.158:8000 > 172.16.112.50:32892: R 0:0(0) ack 1034302996 win 0
1999-04-06 16:06:27.801652 172.16.112.50:32893 > 196.37.75.158:8000: S 1038185830:1038185830(0) win 8760
1999-04-06 16:06:27.802828 196.37.75.158:8000 > 172.16.112.50:32893: S 546626866:546626866(0) ack 1038185831 win 32736
1999-04-06 16:06:27.803026 172.16.112.50:32893 > 196.37.75.158:8000: . ack 546626867 win 8760
1999-04-06 16:06:27.804224 172.16.112.50:32893 > 196.37.75.158:8000: P 1038185831:1038185867(36) ack 546626867 win 8760
1999-04-06 16:06:27.824121 196.37.75.158:8000 > 172.16.112.50:32893: . ack 1038185867 win 32736
```

```
1999-04-06 16:06:27.824475 172.16.112.50:32893 > 196.37.75.158:8000: P 1038185867:1038186069(202) ack 546626867 win 8760
1999-04-06 16:06:27.826642 196.37.75.158:8000 > 172.16.112.50:32893: P 546626867:546626898(31) ack 1038186069 win 32736
1999-04-06 16:06:27.829838 196.37.75.158:8000 > 172.16.112.50:32893: P 546626898:546627507(609) ack 1038186069 win 32736
1999-04-06 16:06:27.829905 196.37.75.158:8000 > 172.16.112.50:32893: F 546627507:546627507(0) ack 1038186069 win 32736
1999-04-06 16:06:27.830054 172.16.112.50:32893 > 196.37.75.158:8000: . ack 546627507 win 8760
1999-04-06 16:06:27.830123 172.16.112.50:32893 > 196.37.75.158:8000: . ack 546627508 win 8760
1999-04-06 16:06:27.915897 172.16.112.50:32894 > 196.37.75.158:8000: S 1038315772:1038315772(0) win 8760
1999-04-06 16:06:27.916915 196.37.75.158:8000 > 172.16.112.50:32894: S 2601713000:2601713000(0) ack 1038315773 win 32736
1999-04-06 16:06:27.917118 172.16.112.50:32894 > 196.37.75.158:8000: . ack 2601713001 win 8760
1999-04-06 16:06:27.918288 172.16.112.50:32894 > 196.37.75.158:8000: P 1038315773:1038315809(36) ack 2601713001 win 8760
1999-04-06 16:06:27.934082 196.37.75.158:8000 > 172.16.112.50:32894: . ack 1038315809 win 32736
1999-04-06 16:06:27.934446 172.16.112.50:32894 > 196.37.75.158:8000: P 1038315809:1038316020(211) ack 2601713001 win 876
1999-04-06 16:06:27.936245 196.37.75.158:8000 > 172.16.112.50:32894: F 2601713001:2601713001(0) ack 1038316020 win 32736
1999-04-06 16:06:27.936379 172.16.112.50:32894 > 196.37.75.158:8000: . ack 2601713002 win 8760
1999-04-06 16:06:27.936955 172.16.112.50:32894 > 196.37.75.158:8000: F 1038316020:1038316020(0) ack 2601713002 win 8760
1999-04-06 16:06:27.937824 196.37.75.158:8000 > 172.16.112.50:32894: . ack 1038316021 win 32735
1999-04-06 16:06:57.939569 172.16.112.50:32895 > 196.37.75.158:8000: S 1042105644:1042105644(0) win 8760
1999-04-06 16:06:57.940615 196.37.75.158:8000 > 172.16.112.50:32895: R 0:0(0) ack 1042105645 win 0
1999-04-06 16:07:27.936274 172.16.112.50:32896 > 196.37.75.158:8000: S 1045939333:1045939333(0) win 8760
1999-04-06 16:07:27.937306 196.37.75.158:8000 > 172.16.112.50:32896: R 0:0(0) ack 1045939334 win 0
1999-04-06 16:07:57.933713 172.16.112.50:32897 > 196.37.75.158:8000: S 1049762779:1049762779(0) win 8760
1999-04-06 16:07:57.934654 196.37.75.158:8000 > 172.16.112.50:32897: R 0:0(0) ack 1049762780 win 0
1999-04-06 16:08:27.930939 172.16.112.50:32898 > 196.37.75.158:8000: S 1053846751:1053846751(0) win 8760
1999-04-06 16:08:27.931992 196.37.75.158:8000 > 172.16.112.50:32898: R 0:0(0) ack 1053846752 win 0
1999-04-06 16:08:57.928273 172.16.112.50:32899 > 196.37.75.158:8000: S 1057776800:1057776800(0) win 8760
1999-04-06 16:08:57.929336 196.37.75.158:8000 > 172.16.112.50:32899: R 0:0(0) ack 1057776801 win 0
1999-04-06 16:09:27.925952 172.16.112.50:32900 > 196.37.75.158:8000: S 1061663855:1061663855(0) win 8760
1999-04-06 16:09:27.926914 196.37.75.158:8000 > 172.16.112.50:32900: R 0:0(0) ack 1061663856 win 0
1999-04-06 16:09:57.923243 172.16.112.50:32893 > 196.37.75.158:8000: F 1038186069:1038186069(0) ack 546627508 win 8760
1999-04-06 16:09:57.924310 196.37.75.158:8000 > 172.16.112.50:32893: R 546627508:546627508(0) win 0

Undocumented: Connection attempt to port 9000 CSlistener
1999-04-06 18:51:16.975569 172.16.112.100:4549 > 209.3.209.166:9000: S 25650750:25650750(0) win 8192
1999-04-06 18:51:16.978625 209.3.209.166:9000 > 172.16.112.100:4549: R 0:0(0) ack 25650751 win 0

Possible Tcpreset attack: low port - low port
1999-04-07 14:21:15.393428 172.16.112.50:123 > 172.16.112.10:123: R 470091505:470091505(0) win 1
1999-04-07 14:22:19.378014 172.16.112.50:123 > 172.16.112.10:123: R 470091505:470091505(0) win 1
1999-04-07 14:23:23.362619 172.16.112.50:123 > 172.16.112.10:123: R 470091505:470091505(0) win 1
1999-04-07 14:24:27.356408 172.16.112.50:123 > 172.16.112.10:123: R 470091505:470091505(0) win 1
1999-04-07 14:25:31.340729 172.16.112.50:123 > 172.16.112.10:123: R 470091505:470091505(0) win 1
1999-04-07 14:26:35.325019 172.16.112.50:123 > 172.16.112.10:123: R 470091505:470091505(0) win 1
1999-04-07 14:27:39.309218 172.16.112.50:123 > 172.16.112.10:123: R 470091505:470091505(0) win 1

NetBus: high port - high port
1999-04-07 16:03:40.582138 206.48.44.18:1290 > 172.16.112.100:12345: S 16062765:16062765(0) win 8192
1999-04-07 16:03:40.582296 172.16.112.100:12345 > 206.48.44.18:1290: S 16057937:16057937(0) ack 16062766 win 8760
1999-04-07 16:03:40.585034 206.48.44.18:1290 > 172.16.112.100:12345: . ack 16057938 win 8760
...
1999-04-07 16:04:35.056918 206.48.44.18:1292 > 172.16.112.100:12346: S 16117250:16117250(0) win 8192
1999-04-07 16:04:35.057080 172.16.112.100:12346 > 206.48.44.18:1292: S 16112421:16112421(0) ack 16117251 win 8760
1999-04-07 16:04:35.057919 206.48.44.18:1292 > 172.16.112.100:12346: . ack 16112422 win 8760

Undocumented: Probe for port 8000 (HTTPtunnel?)
1999-04-07 16:05:25.515349 172.16.112.50:32937 > 196.37.75.158:8000: S 2028664164:2028664164(0) win 8760
1999-04-07 16:05:25.518000 196.37.75.158:8000 > 172.16.112.50:32937: R 0:0(0) ack 2028664165 win 0
1999-04-07 16:05:55.521997 172.16.112.50:32938 > 196.37.75.158:8000: S 2032483304:2032483304(0) win 8760
1999-04-07 16:05:55.522725 196.37.75.158:8000 > 172.16.112.50:32938: R 0:0(0) ack 2032483305 win 0
1999-04-07 16:06:25.519367 172.16.112.50:32939 > 196.37.75.158:8000: S 2036376161:2036376161(0) win 8760
1999-04-07 16:06:25.520056 196.37.75.158:8000 > 172.16.112.50:32939: R 0:0(0) ack 2036376162 win 0
1999-04-07 16:06:55.516587 172.16.112.50:32940 > 196.37.75.158:8000: S 2040299031:2040299031(0) win 8760
1999-04-07 16:06:55.517307 196.37.75.158:8000 > 172.16.112.50:32940: R 0:0(0) ack 2040299032 win 0
1999-04-07 16:07:25.513912 172.16.112.50:32941 > 196.37.75.158:8000: S 2044219338:2044219338(0) win 8760
1999-04-07 16:07:25.514746 196.37.75.158:8000 > 172.16.112.50:32941: R 0:0(0) ack 2044219339 win 0
```

```
1999-04-07 16:07:55.511653 172.16.112.50:32942 > 196.37.75.158:8000: S 2048134582:2048134582(0) win 8760
1999-04-07 16:07:55.512441 196.37.75.158:8000 > 172.16.112.50:32942: R 0:0(0) ack 2048134583 win 0
1999-04-07 16:08:25.508373 172.16.112.50:32943 > 196.37.75.158:8000: S 2052085791:2052085791(0) win 8760
1999-04-07 16:08:25.509102 196.37.75.158:8000 > 172.16.112.50:32943: R 0:0(0) ack 2052085792 win 0
1999-04-07 16:08:55.505699 172.16.112.50:32944 > 196.37.75.158:8000: S 2055974755:2055974755(0) win 8760
1999-04-07 16:08:55.506382 196.37.75.158:8000 > 172.16.112.50:32944: R 0:0(0) ack 2055974756 win 0
1999-04-07 16:09:25.503053 172.16.112.50:32946 > 196.37.75.158:8000: S 2059902500:2059902500(0) win 8760
1999-04-07 16:09:25.503734 196.37.75.158:8000 > 172.16.112.50:32946: R 0:0(0) ack 2059902501 win 0

NTinfoscan: low port - low port
1999-04-08 15:21:28.137220 172.16.112.100:20 > 206.48.44.18:20: S 3273078:3273078(0) win 8192
1999-04-08 15:21:28.138062 206.48.44.18:20 > 172.16.112.100:20: R 0:0(0) ack 3273079 win 0

HTTPtunnel: high port - high port
1999-04-08 16:04:55.437055 172.16.112.50:32935 > 196.37.75.158:8000: S 2009485840:2009485840(0) win 8760
1999-04-08 16:04:55.437833 196.37.75.158:8000 > 172.16.112.50:32935: R 0:0(0) ack 2009485841 win 0
1999-04-08 16:05:25.434275 172.16.112.50:32936 > 196.37.75.158:8000: S 2013381706:2013381706(0) win 8760
1999-04-08 16:05:25.434968 196.37.75.158:8000 > 172.16.112.50:32936: R 0:0(0) ack 2013381707 win 0
1999-04-08 16:05:55.431375 172.16.112.50:32938 > 196.37.75.158:8000: S 2017339930:2017339930(0) win 8760
1999-04-08 16:05:55.432149 196.37.75.158:8000 > 172.16.112.50:32938: R 0:0(0) ack 2017339931 win 0
1999-04-08 16:06:25.429153 172.16.112.50:32939 > 196.37.75.158:8000: S 2021268504:2021268504(0) win 8760
1999-04-08 16:06:25.430145 196.37.75.158:8000 > 172.16.112.50:32939: S 782193537:782193537(0) ack 2021268505 win 32736
1999-04-08 16:06:25.430335 172.16.112.50:32939 > 196.37.75.158:8000: . ack 782193538 win 8760
1999-04-08 16:06:25.431504 172.16.112.50:32939 > 196.37.75.158:8000: P 2021268505:2021268541(36) ack 782193538 win 8760
1999-04-08 16:06:25.443032 196.37.75.158:8000 > 172.16.112.50:32939: . ack 2021268541 win 32736
1999-04-08 16:06:25.443483 172.16.112.50:32939 > 196.37.75.158:8000: P 2021268541:2021268743(202) ack 782193538 win 8760
1999-04-08 16:06:25.445603 196.37.75.158:8000 > 172.16.112.50:32939: P 782193538:782193569(31) ack 2021268743 win 32736
1999-04-08 16:06:25.448532 196.37.75.158:8000 > 172.16.112.50:32939: P 782193569:782194152(583) ack 2021268743 win 32736
1999-04-08 16:06:25.448599 196.37.75.158:8000 > 172.16.112.50:32939: F 782194152:782194152(0) ack 2021268743 win 32736
1999-04-08 16:06:25.448743 172.16.112.50:32939 > 196.37.75.158:8000: . ack 782194152 win 8760
1999-04-08 16:06:25.448811 172.16.112.50:32939 > 196.37.75.158:8000: . ack 782194153 win 8760
1999-04-08 16:06:25.539221 172.16.112.50:32940 > 196.37.75.158:8000: S 2021343082:2021343082(0) win 8760
1999-04-08 16:06:25.540145 196.37.75.158:8000 > 172.16.112.50:32940: S 1449041600:1449041600(0) ack 2021343083 win 32736
1999-04-08 16:06:25.540341 172.16.112.50:32940 > 196.37.75.158:8000: . ack 1449041601 win 8760
1999-04-08 16:06:25.541705 172.16.112.50:32940 > 196.37.75.158:8000: P 2021343083:2021343119(36) ack 1449041601 win 8760
1999-04-08 16:06:25.553131 196.37.75.158:8000 > 172.16.112.50:32940: . ack 2021343119 win 32736
1999-04-08 16:06:25.553462 172.16.112.50:32940 > 196.37.75.158:8000: . ack 1449041601 win 8760
1999-04-08 16:06:25.554693 172.16.112.50:32940 > 196.37.75.158:8000: . ack 1449041601 win 8760
1999-04-08 16:06:25.555620 172.16.112.50:32940 > 196.37.75.158:8000: P 2021344749:2021345796(1047) ack 1449041601 win 87
1999-04-08 16:06:25.567564 196.37.75.158:8000 > 172.16.112.50:32940: F 1449041601:1449041601(0) ack 2021345796 win 32736
1999-04-08 16:06:25.567698 172.16.112.50:32940 > 196.37.75.158:8000: . ack 1449041602 win 8760
1999-04-08 16:06:25.568273 172.16.112.50:32940 > 196.37.75.158:8000: F 2021345796:2021345796(0) ack 1449041602 win 8760
1999-04-08 16:06:25.569135 196.37.75.158:8000 > 172.16.112.50:32940: . ack 2021345797 win 32735
1999-04-08 16:06:55.565921 172.16.112.50:32941 > 196.37.75.158:8000: S 2025130416:2025130416(0) win 8760
1999-04-08 16:06:55.566926 196.37.75.158:8000 > 172.16.112.50:32941: R 0:0(0) ack 2025130417 win 0
1999-04-08 16:07:25.563552 172.16.112.50:32942 > 196.37.75.158:8000: S 2029151076:2029151076(0) win 8760
1999-04-08 16:07:25.566348 196.37.75.158:8000 > 172.16.112.50:32942: R 0:0(0) ack 2029151077 win 0
1999-04-08 16:07:55.570564 172.16.112.50:32944 > 196.37.75.158:8000: S 2033074131:2033074131(0) win 8760
1999-04-08 16:07:55.571535 196.37.75.158:8000 > 172.16.112.50:32944: R 0:0(0) ack 2033074132 win 0
1999-04-08 16:08:25.567783 172.16.112.50:32945 > 196.37.75.158:8000: S 2036884912:2036884912(0) win 8760
1999-04-08 16:08:25.568768 196.37.75.158:8000 > 172.16.112.50:32945: R 0:0(0) ack 2036884913 win 0
1999-04-08 16:08:55.565484 172.16.112.50:32946 > 196.37.75.158:8000: S 2040782008:2040782008(0) win 8760
1999-04-08 16:08:55.566449 196.37.75.158:8000 > 172.16.112.50:32946: R 0:0(0) ack 2040782009 win 0
1999-04-08 16:09:25.562437 172.16.112.50:32947 > 196.37.75.158:8000: S 2044583366:2044583366(0) win 8760
1999-04-08 16:09:25.563463 196.37.75.158:8000 > 172.16.112.50:32947: R 0:0(0) ack 2044583367 win 0
1999-04-08 16:09:55.559783 172.16.112.50:32939 > 196.37.75.158:8000: F 2021268743:2021268743(0) ack 782194153 win 8760
1999-04-08 16:09:55.560884 196.37.75.158:8000 > 172.16.112.50:32939: R 782194153:782194153(0) win 0

SATAN: Portscan high port - high port, sequential destination ports on one host
1999-04-08 18:58:23.079487 209.74.60.168:11240 > 172.16.114.50:1024: S 263678194:263678194(0) win 512
1999-04-08 18:58:23.079683 172.16.114.50:1024 > 209.74.60.168:11240: R 0:0(0) ack 263678195 win 0
...
1999-04-08 18:58:31.419517 209.74.60.168:10168 > 172.16.114.50:9787: S 3116175040:3116175040(0) win 32120
1999-04-08 18:58:31.419722 172.16.114.50:9787 > 209.74.60.168:10168: R 0:0(0) ack 3116175041 win 0
```

```
NTinfoscan: low port - low port
1999-04-08 22:30:50.907190 172.16.112.100:20 > 206.48.44.18:20: S 29040578:29040578(0) win 8192
1999-04-08 22:30:50.908086 206.48.44.18:20 > 172.16.112.100:20: R 0:0(0) ack 29040579 win 0
1999-04-08 22:30:51.422096 172.16.112.100:20 > 206.48.44.18:20: S 29040578:29040578(0) win 8192
1999-04-08 22:30:51.422919 206.48.44.18:20 > 172.16.112.100:20: R 0:0(0) ack 29040579 win 0
1999-04-08 22:30:51.968878 172.16.112.100:20 > 206.48.44.18:20: S 29040578:29040578(0) win 8192
1999-04-08 22:30:51.969796 206.48.44.18:20 > 172.16.112.100:20: R 0:0(0) ack 29040579 win 0
1999-04-08 22:30:52.515649 172.16.112.100:20 > 206.48.44.18:20: S 29040578:29040578(0) win 8192
1999-04-08 22:30:52.516552 206.48.44.18:20 > 172.16.112.100:20: R 0:0(0) ack 29040579 win 0
```

Annex E
Summary of *Sequence* Classifier Results

The *Sequence* classifier uses the TCP flags, TCP sequence number and TCP acknowledgement number attributes to discover:

- SEQ=ACK=0 and not R or RA

```
FIN scan:
1999-04-05 13:43:08.073616 208.240.124.83:43170 > 172.16.112.50:3: F 0:0(0) win 2048
...
1999-04-05 13:46:50.927546 208.240.124.83:62309 > 172.16.112.50:9: F 0:0(0) win 2048

FIN scan:
1999-04-07 16:37:05.119686 204.97.153.43:33731 > 172.16.114.50:1: F 0:0(0) win 3072
1999-04-07 16:37:11.119509 204.97.153.43:33732 > 172.16.114.50:1: F 0:0(0) win 3072
1999-04-07 16:38:11.212840 204.97.153.43:48334 > 172.16.114.50:2: F 0:0(0) win 4096
1999-04-07 16:39:11.271440 204.97.153.43:36206 > 172.16.114.50:3: F 0:0(0) win 1024
1999-04-07 16:40:11.330299 204.97.153.43:34897 > 172.16.114.50:4: F 0:0(0) win 4096
1999-04-07 16:41:11.398469 204.97.153.43:44837 > 172.16.114.50:5: F 0:0(0) win 3072
1999-04-07 16:42:11.476742 204.97.153.43:57319 > 172.16.114.50:6: F 0:0(0) win 4096
1999-04-07 16:43:11.555454 204.97.153.43:42505 > 172.16.114.50:7: F 0:0(0) win 2048
1999-04-07 16:43:17.563378 204.97.153.43:42506 > 172.16.114.50:7: F 0:0(0) win 2048
1999-04-07 16:44:23.701950 204.97.153.43:47885 > 172.16.114.50:8: F 0:0(0) win 4096
1999-04-07 16:45:24.019973 204.97.153.43:47234 > 172.16.114.50:9: F 0:0(0) win 4096
1999-04-07 16:45:30.025698 204.97.153.43:47235 > 172.16.114.50:9: F 0:0(0) win 4096
1999-04-07 16:46:36.147226 204.97.153.43:53912 > 172.16.114.50:10: F 0:0(0) win 4096
```

Annex F
ACRONYMS

ANN	Artificial Neural Network
CGI	Common Gateway Interface
DARPA	Defense Advanced Research Projects Agency
DDoS	Distributed Denial of Service
DoS	Denial of Service
DNS	Domain Name System
DREnet	Defence Research Establishment Network
ECN	Explicit Congestion Control
FQDN	Fully Qualified Domain Name
FTP	File Transfer Protocol
HTML	HyperText Markup Language
HTRQ	Hypertext Request
HTTP	Hypertext Transfer Protocol
ID	Intrusion Detection
IDS	Intrusion Detection System
IP	Internet Protocol
ISN	The Initial Sequence Number
IPS	Intrusion Prevention System
MSE	Mean Square Error
NE	Network Element
NTA	Network Traffic Analysis
OS	Operating System
SMTP	Simple Mail Transfer Protocol
TCP	Transmission Control Protocol
UDP	User Datagram Protocol
UI	User Interface
URL	Uniform Resource Locator
WWW	World Wide Web

DOCUMENT CONTROL DATA

(Security classification of title, body of abstract and indexing annotation must be entered when the overall document is classified)

1. ORIGINATOR (the name and address of the organization preparing the document. Organizations for whom the document was prepared, e.g. Establishment sponsoring a contractor's report, or tasking agency, are entered in section 8.)

Defence R&D Canada – Ottawa
Ottawa ON K1A 0Z4

2. SECURITY CLASSIFICATION (overall security classification of the document, including special warning terms if applicable)

UNCLASSIFIED

3. TITLE (the complete document title as indicated on the title page. Its classification should be indicated by the appropriate abbreviation (S,C or U) in parentheses after the title.)

Investigation of a Neural Network Implementation of a TCP Packet Anomaly Detection System (U)

4. AUTHORS (Last name, first name, middle initial)

Dondo Maxwell G and Treurniet Joanne R

5. DATE OF PUBLICATION (month and year of publication of document)	6a. NO. OF PAGES (total containing information. Include Annexes, Appendices, etc.)	6b. NO. OF REFS (total cited in document)
May 2004	53	34

7. DESCRIPTIVE NOTES (the category of the document, e.g. technical report, technical note or memorandum. If appropriate, enter the type of report, e.g. interim, progress, summary, annual or final. Give the inclusive dates when a specific reporting period is covered.)

Technical Memorandum

8. SPONSORING ACTIVITY (the name of the department project office or laboratory sponsoring the research and development. Include the address.)

NIO

9a. PROJECT OR GRANT NO. (if appropriate, the applicable research and development project or grant number under which the document was written. Please specify whether project or grant)	9b. CONTRACT NO. (if appropriate, the applicable number under which the document was written)

10a. ORIGINATOR'S DOCUMENT NUMBER (the official document number by which the document is identified by the originating activity. This number must be unique to this document.)	10b. OTHER DOCUMENT NOS. (Any other numbers which may be assigned this document either by the originator or by the sponsor)
DRDC Ottawa TM 2004-208	

11. DOCUMENT AVAILABILITY (any limitations on further dissemination of the document, other than those imposed by security classification)

(X) Unlimited distribution
( ) Distribution limited to defence departments and defence contractors; further distribution only as approved
( ) Distribution limited to defence departments and Canadian defence contractors; further distribution only as approved
( ) Distribution limited to government departments and agencies; further distribution only as approved
( ) Distribution limited to defence departments; further distribution only as approved
( ) Other (please specify):

12. DOCUMENT ANNOUNCEMENT (any limitation to the bibliographic announcement of this document. This will normally correspond to the Document Availability (11). However, where further distribution (beyond the audience specified in 11) is possible, a wider announcement audience may be selected.)

13. ABSTRACT (a brief and factual summary of the document. It may also appear elsewhere in the body of the document itself. It is highly desirable that the abstract of classified documents be unclassified. Each paragraph of the abstract shall begin with an indication of the security classification of the information in the paragraph (unless the document itself is unclassified) represented as (S), (C), or (U). It is not necessary to include here abstracts in both official languages unless the text is bilingual).

We present the design and implementation of an artificial neural network (ANN) system of multi-layer perceptron classifiers to detect suspicious TCP traffic at a single packet level. The advantage to using ANNs for the detection of attacks is that they do not only rely on attack signatures, as in many common signature-based IDSs. Rather they are capable of learning broader definitions of attack attributes. The use of ANNs in this approach also enhances the processing speed where real-time applications require the processing of substantial amounts of data at high speeds. The ANN model was tested on labelled sets of attack data obtained from the DARPA IDS Evaluation. The model was successful in detecting a variety of attacks, including denial of service attacks, probing activity and other suspicious activity. Future work will examine the application of an ANN to sequences of related packets to detect attacks.

14. KEYWORDS, DESCRIPTORS or IDENTIFIERS (technically meaningful terms or short phrases that characterize a document and could be helpful in cataloguing the document. They should be selected so that no security classification is required. Identifiers such as equipment model designation, trade name, military project code name, geographic location may also be included. If possible keywords should be selected from a published thesaurus, e.g. Thesaurus of Engineering and Scientific Terms (TEST) and that thesaurus-identified. If it is not possible to select indexing terms which are Unclassified, the classification of each should be indicated as with the title.)

Intrusion detection, anomaly detection, TCP/IP

Defence R&D Canada

Canada's leader in defence
and national security R&D

R & D pour la défense Canada

Chef de file au Canada en R & D
pour la défense et la sécurité nationale

www.drdc-rddc.gc.ca